

Course Code: BCA 105

Practical Workbook

Course Title: Lab – I

C Programming Language

Name	:	_____
Year	:	_____
Batch	:	_____
Roll No	:	_____
Department:		_____
Teacher :		_____

**Department of Computer Applications
F.Y.B.C.A Science (SEM I)**

Savitribai Phule Pune University, Pune

INTRODUCTION

The Practical Workbook for “Programming Languages” introduces the basic as well as advance concepts of programming using C language. C encompasses the characteristics of both the high level languages; which give a better programming efficiency and faster program development; and the low level languages; which have a better machine efficiency.

Each lab session begins with a brief theory of the topic. Many details have not been incorporated as the same is to be covered in Theory classes. The Exercise section follows this section.

The Workbook has been arranged as seventeen labs starting with a practical on the Introduction to programming environment and fundamentals of programming language.

Next lab session deals with single stepping; an efficient debugging and error detection technique.

Next three lab sessions cover the basic building blocks of programming. These practicals introduce the concepts of decision making; loops; function declaration and definition etc.

The next three experiments deal with the advance concepts like arrays, pointers, structures and unions. These features enable the users to handle not only large amount of data, but also data of different types (integers, characters etc.) and to do so efficiently.

Separate practical's have been included for different graphics and text modes, passing variable number of arguments to functions, and command line arguments.

Further two lab sessions covers the filing feature, which allows the user to store/retrieve the data on/from permanent storage like floppy or hard disk and various file and directory manipulation functions.

Finally, there are labs on hardware interfacing using ROM BIOS routines, which explain accessing the system color palettes, interfacing mouse etc. in programs using C.

Few changes have been made in the examples and exercises of this workbook since its last edition which was printed in 2004. Now these are more comprehensive than those of the previous issue. One lab session of the previous workbook is replaced by an appendix that discusses various date and time functions.

CONTENTS

Sr. No.	Assignment	No. of Slots
1.	Familiarization with Programming Environment using Turbo C and Fundamentals of Programming Language	01
2.	Assignment on use of data types, simple operators (expressions)	02
3.	Debugging and Single-Stepping of Programs	01
4.	Assignment on decision making statements (if and if-else, nested structures)	02
5.	Assignment on decision making statements (switch case)	01
6.	Assignment on use of while loops	02
7.	Assignment on use of for loops	02
8.	Assignment on nested loops	01
9.	Assignment on exit, goto, continue, break	02
10.	Assignment on menu driven programs.	02
11.	Assignment on writing C programs in modular way (use of user defined functions)	02
12.	Assignment on call by value	01
13.	Assignment on call by reference	01
14.	Assignment on recursive functions	02
15.	Assignment on use of arrays (1-D array) and functions	02
16.	Assignment on use of multidimensional array (2-D arrays) and functions	02
17.	Assignment on Standard Library Function	02
	Total slots Required	28

Lab Session 01

OBJECT

Familiarization with Programming Environment using Turbo C and Fundamentals of Programming Language

THEORY

The Development Environment - Integrated Development Environment (IDE):

The C compiler has its own built-in text editor. You may also use a commercial text editor or word processor that can produce text files. The important thing is that whatever you write your program in, it must save simple, plain-text files, with no word processing commands embedded in the text. The files you create with your editor are called source files, and for C++ they typically are named with the extension *.CPP*, *.CP*, or *.C*.

The C Developing Environment, also called as Programmers' Platform, is a screen display with windows and pull-down menus. Code of the program, error messages and other information are displayed in separate windows. The menus may be used to invoke the operations necessary to develop the program, debug and execute the program.

Invoking the IDE

To invoke the IDE from the windows you need to double click the TC icon.

To do so from the command prompt go in the specific directory and type 'tc'. This makes you enter the IDE interface, which initially displays only a menu bar at the top of the screen and a status line below will appear. The menu bar displays the menu names and the status line tells what various function keys will do.

Using Menus

If the menu bar is inactive, it may be invoked by pressing the [F10] function key. To select different menu, move the highlight left or right with cursor (arrow) keys. You can also revoke the selection by pressing the key combination for the specific menu.

Opening New Window

To type a program, you need to open an Edit Window. For this, open file menu and click “new”. A window will appear on the screen where the program may be typed.

Writing a Program

When the Edit window is active, the program may be typed. Use the certain key combinations to perform specific edit functions.

Saving a Program

To save the program, select **save** command from the **file** menu. This function can also be performed by pressing the [F2] button. A dialog box will appear asking for the path and name of the file. Provide an appropriate and unique file name. You can save the program after compiling too but saving it before compilation is more appropriate.

Making an Executable File

The source file is required to be turned into an executable file. This is called “Making” of the

.exe file. The steps required to create an executable file are:

1. Create a source file, with a .c extension.
2. Compile the source code into a file with the .obj extension.
3. Link your .obj file with any needed libraries to produce an executable program.

Compiling the Source Code

Although the source code in your file is somewhat cryptic, and anyone who doesn't know C will struggle to understand what it is for, it is still in what we call human-readable form. But, for the computer to understand this source code, it must be converted into machine-readable form. This is done by using a compiler. Hence, compiling is the process in which source code is translated into machine understandable language.

Creating an Executable File with the Linker

After your source code is compiled, an object file is produced. This file is often named with the extension .OBJ. This is still not an executable program, however. To turn this into an executable program, you must run your linker. C programs are typically created by linking together one or more OBJ files with one or more libraries. A library is a collection of linkable files that were supplied with your compiler.

Project/Make

Before compiling and linking a file, a part of the IDE called Project/Make checks the time and date on the file you are going to compile.

Compiling and linking in the IDE

In the Turbo C IDE, compiling and linking can be performed together in one step. There are two ways to do this: you can select Make EXE from the compile menu, or you can press the [F9] key.

Executing a Program

If the program is compiled and linked without errors, the program is executed by selecting Run from the Run Menu or by pressing the [Ctrl+F9] key combination.

The Development Cycle

If every program worked the first time you tried it, that would be the complete development cycle: Write the program, compile the source code, link the program, and run it. Unfortunately, almost every program, no matter how trivial, can and will have errors, or bugs, in the program. Some bugs will cause the compile to fail, some will cause the link to fail, and some will only show up when you run the program. Whatever type of bug you find, you must fix it, and that involves editing your source code, recompiling and relinking, and then rerunning the program.

Correcting Errors

If the compiler recognizes some error, it will let you know through the Compiler window. You'll see that the number of errors is not listed as 0, and the word "Error" appears instead of the word "Success" at the bottom of the window. The errors are to be removed by returning to the edit window. Usually these errors are a result of a typing mistake. The compiler will not only tell you what you did wrong; they'll point you to the exact place in your code where you made the mistake.

Exiting IDE

An Edit window may be closed in a number of different ways. You can click on the small square in the upper left corner, you can select **close** from the **window** menu, or you can press the [Alt][F3] combination. To exit from the IDE select **Exit** from the **File** menu or press [Alt][X] combination.

C Program Structure – First C Program

A C program source code can be written in any text editor; however the file should be saved with .c extension. Lets write the First C program.

STEPS TO WRITE C PROGRAMS AND GET THE OUTPUT:

Below are the steps to be followed for any C program to create and get the output. This is common to all C program and there is no exception whether its a very small C program or very large C program.

1. Create
2. Compile
3. Execute or Run
4. Get the Output

CREATION, COMPILATION AND EXECUTION OF A C PROGRAM:

Prerequisite:

- If you want to create, compile and execute C programs by your own, you have to install C compiler in your machine. Then, you can start to execute your own C programs in your machine.
- You can refer below link for how to install C compiler and compile and execute C programs in your machine.
- Once C compiler is installed in your machine, you can create, compile and execute C programs as shown in below link.
- If you don't want to install C/C++ compilers in your machine, you can refer online compilers which will compile and execute C/C++ and many other programming languages online and display outputs on the screen. Please search for online C/C++ compilers in Google for more details.

ASIC STRUCTURE OF A C PROGRAM:

Structure of C program is defined by set of rules called protocol, to be followed by programmer while writing C program. All C programs are having sections/parts which are mentioned below.

1. Documentation section
2. Link Section
3. Definition Section
4. Global declaration section
5. Function prototype declaration section
6. Main function
7. User defined function definition section

EXAMPLE C PROGRAM TO COMPARE ALL THE SECTIONS:

You can compare all the sections of a C program with the below C program.

```

1  /*
2   Documentation section
3   C programming basics & structure of C programs
4   Author: New Author
5   Date :01/01/2018
6  */
7
8  #include <stdio.h> /* Link section */
9  int total = 0;      /* Global declaration, definition section */
10 int sum (int, int); /* Function declaration section */
11 int main ()         /* Main function */
12 {
13     printf ("This is a C basic program \n");
14     total = sum (1, 1);
15     printf ("Sum of two numbers : %d \n", total);
16     return 0;
17 }
18
19 int sum (int a, int b) /* User defined function */
20 {
21     return a + b;      /* definition section */
22 }

```

OUTPUT:

```

This is a C basic program
Sum of two numbers : 2

```

DESCRIPTION FOR EACH SECTION OF THE C PROGRAM:

- Let us see about each section of a C basic program in detail below.
- Please note that a C program mayn't have all below mentioned sections except main function and link sections.
- Also, a C program structure mayn't be in below mentioned order.

Sections	Description
Documentation section	We can give comments about the program, creation or modified date, author name etc in this section. The characters or words or anything which are given between “/*” and “*/”, won't be considered by C compiler for compilation process. These will be ignored by C compiler

	during compilation. Example : /* comment line1 comment line2 comment 3 */
Link Section	Header files that are required to execute a C program are included in this section
Definition Section	In this section, variables are defined and values are set to these variables.
Global declaration section	Global variables are defined in this section. When a variable is to be used throughout the program, can be defined in this section.
Function prototype declaration section	Function prototype gives many information about a function like return type, parameter names used inside the function.
Main function	Every C program is started from main function and this function contains two major sections called declaration section and executable section.
User defined function section	User can define their own functions in this section which perform particular task as per the user requirement.

A SIMPLE C PROGRAM:

Below C program is a very simple and basic program in C programming language. This C program displays “Hello World!” in the output window. And,

all syntax and commands in C programming are case sensitive. Also, each statement should be ended with semicolon (;) which is a statement terminator.

```
1 #include <stdio.h>
2 int main()
3 {
4     /* Our first simple C basic program */
5     printf("Hello World! ");
6     getch();
7     return 0;
8 }
```

OUTPUT:

Hello World!

Lab Session 02

OBJECT:- Understanding data types and simple operators

C Keywords – Reserved Words

In C, we have 32 keywords, which have their predefined meaning and cannot be used as a variable name. These words are also known as “reserved words”. It is good practice to avoid using these keywords as variable name. These are –

<u>C KEYWORDS OR RESERVED WORDS</u>			
auto	break	case	char
const	continue	default	do
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while
double	else	enum	extern
float	for	goto	if

Basics usage of these keywords –

if, else, switch, case, default – Used for decision control programming structure.

break – Used with any loop OR switch case.

int, float, char, double, long – These are the data types and used during variable declaration.

for, while, do – types of loop structures in C.

void – One of the return type.

goto – Used for redirecting the flow of execution.

auto, signed, const, extern, register, unsigned – defines a variable.

return – This keyword is used for returning a value.

continue – It is generally used with for, while and dowhile loops, when compiler encounters this statement it performs the next iteration of the loop, skipping rest of the statements of current iteration.

enum – Set of constants.

sizeof – It is used to know the size.

struct, typedef – Both of these keywords used in structures (Grouping of data types in a single record).

union – It is a collection of variables, which shares the same memory location and memory storage.

Building Blocks of Programming Language:

In any language there are certain building blocks:

- Constants
- Variables
- Operators
- Methods to get input from user (scanf(), getch() etc.)
- Methods to display output (Format Specifier, Escape Sequences etc.) and so on.

Variables and Constants

If the value of an item can be changed in the program then it is a variable. If it will not change then that item is a constant. The various

variable types (also called *data type*) in C are: *int*, *float*, *char*, *long* etc. For constants, the keyword **const** is added before declaration.

Operators

There are various types of operators that may be placed in following categories:

I. Arithmetic Operators

II. Increment and Decrement Operators

III. Assignment Operators

IV. Relational Operators

V. Logical Operators

VI. Conditional Operators

VII. Bitwise Operators

VIII . Special Operators

I. Arithmetic Operators

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	Multiplication
/	division
%	remainder after division(modulo division)

II. Increment and Decrement Operators

Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in C programs.

Syntax:

Increment operator: ++var_name; (or) var_name++;

Decrement operator: --var_name; (or) var_name --;

Example:

Increment operator : ++ i ; i ++ ;

Decrement operator : -- i ; i -- ;

Difference between pre/post increment & decrement operators in C:

Below table will explain the difference between pre/post increment and decrement operators in C programming language

Operator	Operator Description
Pre increment operator (++i)	value of i is incremented before assigning it to the variable i
Post increment operator (i++)	value of i is incremented after assigning it to the variable i
Pre decrement operator (--i)	value of i is decremented before assigning it to the variable I
Post decrement operator (i--)	value of i is decremented after assigning it to variable i

Example program for pre – increment operators in C:

//Example for increment operators

```
#include <stdio.h>
int main()
{
    int i=0;
    while(++i < 5 )
    {
        printf("%d ",i);
    }
    return 0;
}
```

Output:

1 2 3 4

Step 1 : In above program, value of “i” is incremented from 0 to 1 using pre-increment operator.

Step 2 : This incremented value “1” is compared with 5 in while expression.

Step 3 : Then, this incremented value “1” is assigned to the variable “i”. Above 3 steps are continued until while expression becomes false and output is displayed as “1 2 3 4”.

Example program for post – increment operators in C:

```
#include <stdio.h>
int main()
{
    int i=0;
```



```

        while(i++ < 5 )
        {
            printf("%d ",i);
        }
        return 0;
}

```

Output:

1 2 3 4 5

Step 1 : In this program, value of i “0” is compared with 5 in while expression.

Step 2 : Then, value of “i” is incremented from 0 to 1 using post-increment operator.

Step 3 : Then, this incremented value “1” is assigned to the variable “i”. Above 3 steps are continued until while expression becomes false and output is displayed as “1 2 3 4 5”.

Example program for pre – decrement operators in C:

```

#include <stdio.h>
int main()
{
    int i=10;
    while(--i > 5 )
    {
        printf("%d ",i);
    }
    return 0;
}

```

Output:

9 8 7 6

Step 1 : In above program, value of “i” is decremented from 10 to 9 using pre-decrement operator.

Step 2 : This decremented value “9” is compared with 5 in while expression.

Step 3 : Then, this decremented value “9” is assigned to the variable “i”. Above 3 steps are continued until while expression becomes false and output is displayed as “9 8 7 6”.

Example program for post – decrement operators in C:

```

#include <stdio.h>
int main()
{
    int i=10;
    while(i-- > 5 )
    {
        printf("%d ",i);
    }
    return 0;
}

```

Output:

9 8 7 6 5

Step 1 : In this program, value of i “10” is compared with 5 in while expression.

Step 2 : Then, value of “i” is decremented from 10 to 9 using post-decrement operator.

Step 3 : Then, this decremented value “9” is assigned to the variable “i”. Above 3 steps are continued until while expression becomes false and output is displayed as “9 8 7 6 5”.

III. Assignment Operators

In C programs, values for the variables are assigned using assignment operators.

For example, if the value “10” is to be assigned for the variable “sum”, it can be assigned as “sum = 10;”

There are 2 categories of assignment operators in C language. They are,

1. Simple assignment operator (Example: =)
2. Compound assignment operators (Example: +=, -=, *=, /=, %=, &=, ^=)

Operators	Example/Description
=	sum = 10; 10 is assigned to variable sum
+=	sum += 10; This is same as sum = sum + 10
-=	sum -= 10; This is same as sum = sum - 10
*=	sum *= 10; This is same as sum = sum * 10
/=	sum /= 10; This is same as sum = sum / 10
%=	sum %= 10; This is same as sum = sum % 10

&=	sum&=10; This is same as sum = sum & 10
^=	sum ^= 10; This is same as sum = sum ^ 10

IV. Relational Operators

Relational operators are used to find the relation between two variables. i.e. to compare the values of two variables in a C program

Operators	Example/Description
>	x > y (x is greater than y)
<	x < y (x is less than y)
>=	x >= y (x is greater than or equal to y)
<=	x <= y (x is less than or equal to y)
==	x == y (x is equal to y)
!=	x != y (x is not equal to y)

Example program for relational operators in C:

In this program, relational operator (==) is used to compare 2 values whether they are equal or not.

If both values are equal, output is displayed as "values are equal". Else, output is displayed as "values are not equal".

Note : double equal sign (==) should be used to compare 2 values. We should not single equal sign (=).

```
#include <stdio.h>

int main()
{
```

```

int m=40,n=20;
if (m == n)
{
    printf("m and n are equal");
}
else
{
    printf("m and n are not equal");
}
}
Output:
m and n are not equal

```

V. Logical Operators

These operators are used to perform logical operations on the given expressions.

There are 3 logical operators in C language. They are, logical AND (&&), logical OR (||) and logical NOT (!).

Operators	Example/Description
&& (logical AND)	(x>5)&&(y<5) It returns true when both conditions are true
(logical OR)	(x>=10) (y>=10) It returns true when at-least one of the condition is true
! (logical NOT)	!((x>5)&&(y<5)) It reverses the state of the operand “((x>5) && (y<5))” If “((x>5) && (y<5))” is true, logical NOT operator makes it false

VI. Conditional Operators

CONDITIONAL OR TERNARY OPERATORS IN C:

Conditional operators return one value if condition is true and returns another value if condition is false.

This operator is also called as ternary operator.

Syntax : (Condition? true_value: false_value);

Example : (A > 100 ? 0 : 1);

In above example, if A is greater than 100, 0 is returned else 1 is returned. This is equal to if else conditional statement

Example program for conditional/ternary operators in C:

```
#include <stdio.h>
```

```

int main()
{
    int x=1, y ;
    y = ( x ==1 ? 2 : 0 ) ;
    printf("x value is %d\n", x);
    printf("y value is %d", y);
}

```

Output:

x value is 1

y value is 2

VII. Bitwise Operators

These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise NOT), ^ (XOR), << (left shift) and >> (right shift).

TRUTH TABLE FOR BIT WISE OPERATION & BIT WISE OPERATORS:

x	y	x y	x&y	x^y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Below are the bit-wise operators and their name in C language.

& – Bitwise AND

| – Bitwise OR

~ – Bitwise NOT

^ – XOR

<< – Left Shift

>> – Right Shift

Consider x=40 and y=80. Binary form of these values are given below.

x = 00101000

y= 01010000

All bit wise operations for x and y are given below.

x&y = 00000000 (binary) = 0 (decimal)

x|y = 01111000 (binary) = 120 (decimal)

~x = 111111111111111111111111

11111111111111111111111111111111010111 = -41 (decimal)

x^y = 01111000 (binary) = 120 (decimal)

x << 1 = 01010000 (binary) = 80 (decimal)

x >> 1 = 00010100 (binary) = 20 (decimal)

Note:

Bit wise NOT : Value of 40 in binary is 000000000000000000000000000000
00.
So, all 0's are converted into 1's in bit wise NOT operation.

Bit wise left shift and right shift : In left shift operation “ $x \ll 1$ ”, 1 means that the bits will be left shifted by one place. If we use it as “ $x \ll 2$ ”, then, it means that the bits will be left shifted by 2 places.

Example program for bit wise operators in C:

In this example program, bit wise operations are performed as shown above and output is displayed in decimal format.

```
#include <stdio.h>
```

```
int main()
{
    int m = 40,n = 80,AND_opr,OR_opr,XOR_opr,NOT_opr ;
    AND_opr = (m&n);
    OR_opr = (m | n);
    NOT_opr = (~m);
    XOR_opr = (m^ n);
    printf("AND_opr value = %d\n",AND_opr );
    printf("OR_opr value = %d\n",OR_opr );
    printf("NOT_opr value = %d\n",NOT_opr );
    printf("XOR_opr value = %d\n",XOR_opr );
    printf("left_shift value = %d\n", m << 1);
    printf("right_shift value = %d\n", m >> 1);
}
```

Output:

```
AND_opr value = 0
OR_opr value = 120
NOT_opr value = -41
XOR_opr value = 120
left_shift value = 80
right_shift value = 20
```

VIII . Special Operators

Below are some of the special operators that the C programming language offers.

Operators	Description
&	This is used to get the address of the variable. Example : &a will give address of a.
*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.

printf format identifiers.

These identifiers actually have upto 6 parts as shown in the table below. They MUST be used in the order shown.

%	Flags	Minimum field width	Period	Precision. Maximum field width	Argument type
Required	Optional	Optional	Optional	Optional	Required

Format Specifiers

Format Specifiers tell the *printf* statement where to put the text and how to display the text.

The various format specifiers are:

Format specifier	Description	Supported data types
%c	Character	char unsigned char
%d	Signed Integer	short unsigned short int long

<code>%e</code> or <code>%E</code>	Scientific notation of float values	<code>float</code> <code>double</code>
<code>%f</code>	Floating point	<code>float</code>
<code>%g</code> or <code>%G</code>	Similar as <code>%e</code> or <code>%E</code>	<code>float</code> <code>double</code>
<code>%hi</code>	Signed Integer(Short)	<code>short</code>
<code>%hu</code>	Unsigned Integer(Short)	<code>unsigned short</code>
<code>%i</code>	Signed Integer	<code>short</code> <code>unsigned short</code> <code>int</code> <code>long</code>
<code>%l</code> or <code>%ld</code> or <code>%li</code>	Signed Integer	<code>long</code>
<code>%lf</code>	Floating point	<code>double</code>
<code>%Lf</code>	Floating point	<code>long double</code>
<code>%lu</code>	Unsigned integer	<code>unsigned int</code> <code>unsigned long</code>
<code>%lli</code> , <code>%lld</code>	Signed Integer	<code>long long</code>
<code>%llu</code>	Unsigned Integer	<code>unsigned long long</code>
<code>%o</code>	Octal representation of Integer.	<code>short</code> <code>unsigned short</code> <code>int</code> <code>unsigned int</code> <code>long</code>
<code>%p</code>	Address of pointer to void void *	<code>void *</code>
<code>%s</code>	String	<code>char *</code>
<code>%u</code>	Unsigned Integer	<code>unsigned int</code> <code>unsigned long</code>
<code>%x</code> or <code>%X</code>	Hexadecimal representation of Unsigned Integer	<code>short</code> <code>unsigned short</code>

		int unsigned int long
%n	Prints nothing	
%%	Prints % character	

Flags

The format identifiers can be altered from their default function by applying the following **flags**:

- Left justify.
- 0 Field is padded with 0's instead of blanks.
- + Sign of number always O/P.
- blank Positive values begin with a blank.
- # Various uses:
 - %#o (Octal) 0 prefix inserted.
 - %#x (Hex) 0x prefix added to non-zero values.
 - %#X (Hex) 0X prefix added to non-zero values.
 - %#e Always show the decimal point.
 - %#E Always show the decimal point.
 - %#f Always show the decimal point.
 - %#g Always show the decimal point trailing zeros not removed.
 - %#G Always show the decimal point trailing zeros not removed.

Field Width Specifiers

this is "n is printed right-justified in a field width of 5". "Right-justify" means that the number is placed as far right as possible in the field and spaces added in front of it to make up the field width. If the number is placed as far left as possible and spaces are added after it to make up the field width, the number is left-justified. The minus sign can be used to specify left-justification;

They are used with % to limit precision in floating point number. The number showing limit follows the radix point.

Field Width Example

A field width is useful when we want to line up numbers one below the other. Suppose we have three int variables a, b and c with values 9876, -3 and 501, respectively. The statements

```
printf("%d\n", a);
printf("%d\n", b);
```

```
printf("%d\n", c);  
output will print  
9876  
-3  
501
```

Each number is printed using just the number of columns required. Since this varies from one number to the next, they do not line up. If we want to, we could get the numbers lined up using a field width of 5, say. The statements

```
printf("%5d\n", a);  
printf("%5d\n", b);  
printf("%5d\n", c);
```

output will print (◇ denotes a space)

```
◇9876  
◇◇◇-3  
◇◇501
```

Escape Sequences

Escape Sequence causes the program to **escape** from the normal interpretation of a string, so that the next character is recognized as having a special meaning. The back slash “\” character is called the **Escape Character**. The escape sequence includes the following:

\n	=>	new line
\b	=>	back space
		carriage
\r	=>	return
		double
\”	=>	quotations
		back slash
\\	=>	etc.

Getting Input From the User

The input from the user can be taken by the following techniques: scanf(), getch(), getche(), getchar() etc.

Examples

1. Implementing a Simple C Program

```
#include<conio.h>  
#include<stdio.h>  
void main(void)  
{  
clrscr();  
printf("\n Hello World");  
getch();  
}
```

2. Demonstrating the fundamentals of C Language

```
#include<conio.h>
#include<stdio.h>
void main(void)
{
    clrscr();
    int num1,num2,sum,product;
    printf("\tThe program takes two numbers as input and
           prints their sum and product");
    printf("\n Enter first number:");
    scanf("%d",&num1);
    printf("\n Enter second number:");
    scanf("%d",&num2);
    sum=num1+num2;
    product=num1*num2;
    printf("\n%d+%d=%d",num1,num2,sum);
    printf("\n%d*%d=%d",num1,num2,product);
    getch();
}
```

EXERCISES

1. Type the following program in C Editor and execute it. Mention the Error (if any).

```
void main(void)
{
    printf("This is my first program in C");
}
```

2. Add the following line at the beginning of the above program. Recompile the program. What is the output?

```
#include<stdio.h>
```

3. Make the following changes to the program. Mention the Errors observed, in your own words:

i. Write Void instead of void.

ii. Remove the semi colon ';'.

iii. Erase any one of brace '{' or '}' .

-
4. Write a program to calculate the Area ($A = \pi r^2$) and circumference of a circle ($C = 2\pi r$), where r = radius is taken as input and π is declared as a constant. Display the result to 4 decimal places.

-
5. Write a single C statement to output the following on the screen:

My name is "*Your Name*"
And my roll number is "*00Your_roll_no*"
I am a student of "B.C.A. Science"

SET A

- 1. WAP to find sum of two commands**
- 2. WAP to find simple interest**
- 3. WAP to convert temperature from degree centigrade to Fahrenheit**
- 4. WAP to calculate sum of 5 subjects and find percentage**
- 5. WAP to find gross salary**
 - D.A= 10 % of basic**
 - T.A= 12 % basic**
 - Gross salary = basic+ D.A.+ T.A**

SET B

- 1. WAP to swap two integers –**
 - i) Using temporary variable**
 - ii) Without Using temporary variable**
 - a) Using + and – operator**
 - b) Using / and ***

Lab Session 03

OBJECT

Debugging and Single-Stepping of Programs

THEORY

One of the most innovative and useful features of Turbo C++ is the integration of debugging facilities into the IDE.

Even if your program compiles perfectly, it still may not work. Such errors that cause the program to give incorrect results are called **Logical Errors**. The first thing that should be done is to review the listing carefully. Often, the mistake will be obvious. But, if it is not, you'll need the assistance of the Turbo C Debugger.

One Step at a Time

The first thing that the debugger can do for you is slow down the operation of the program. One trouble with finding errors is that a typical program executes in a few milliseconds, so all you can see is its final state. By invoking C++'s single-stepping capability, you can execute just one line of the program at a time. This way you can follow where the program is going.

Consider the following program:

```
void main(void)
{
    int number, answer=-1;
    number = -50;
    if(number < 100)
        if(number > 0)
            answer = 1;
    else
        answer = 0;
    printf("answer is %d\n", answer);
}
```

Our intention in this program is that when **number** is between 0 and 100, **answer** will be 1, when the **number** is 100 or greater, **answer** will be 0, and when **number** is less than 0, **answer** will retain its initialized value of -1. When we run this program with a test value of -50 for **number**, we find that **answer** is set to 0 at the end of the program, instead of staying -1.

We can understand where the problem is if we single step through the program. To do this, simply press the [F7] key. The first line of the program will be highlighted. This highlighted line is called the **run bar**. Press [F7] again. The run bar will move to the next program line. The run bar appears on the line about to be executed. You can execute each

line of the program in turn by pressing [F7]. Eventually you'll reach the first **if** statement:

```
if (num < 100 )
```

This statement is true (since **number** is -50); so, as we would expect the run bar moves to the second **if** statement:

```
if( num > 0)
```

This is false. Because there's no **else** matched with the second **if**, we would expect the run bar to the **printf()** statement. But it doesn't! It goes to the line

```
answer = 0;
```

Now that we see where the program actually goes, the source of the bug should become clear. The **else** goes with the last **if**, not the first **if** as the indenting would lead us to believe. So, the **else** is executed when the second **if** statement is false, which leads to erroneous results. We need to put braces around the second **if**, or rewrite the program in some other way.

Resetting the Debugger

Suppose you've single stepped part way through a program, and want to start over at the beginning. How do you place the run bar at the top of the listing? You can reset the debugging process and initialize the run bar by selecting the Program Reset option from the Run menu.

Watches

Single stepping is usually used with other features of the debugger. The most useful of these is the watch (or watch expression). This lets you see how the value of variable changes as the program runs. To add a watch expression, press [Ctrl+F7] and type the expression.

Breakpoints

It often happens that you've debugged part of your program, but must deal with a bug in another section, and you don't want to single-step through all the statements in the first part to get to the section with the bug. Or you may have a loop with many iterations that would be tedious to step through. The way to do this is with a breakpoint. A breakpoint marks a statement where the program will stop. If you start the program with [Ctrl][F9], it will execute all the statements up to the breakpoint, then stop. You can now examine the state of the variables at that point using the watch window.

Installing breakpoints

To set a breakpoint, first position the cursor on the appropriate line. Then select Toggle Breakpoint from the Debug menu (or press [Ctrl][F8]). The line with the breakpoint will be highlighted. You can install as many breakpoints as you want. This is useful if the program

can take several different paths, depending on the result of if statements or other branching constructs.

Removing Breakpoints

You can remove a single breakpoint by positioning the cursor on the line with the breakpoint and selecting Toggle breakpoint from the Debug menu or pressing the [Ctrl][F8] combination (just as you did to install the breakpoint). The breakpoint highlight will vanish.

You can all set **Conditional Breakpoints** that would break at the specified value only.

EXERCISES

1. Fill out all the entities in table by their corresponding values by inserting watches and single stepping the program.

i. int num=*your_roll_no*, sqr ;
sqr= num*num ;

Before Execution		After Execution	
num	sqr	num	sqr

ii. char ch='First letter of your Name'; ch
++;

Before Execution		After Execution	
ch		ch	

iii. int x=*your_roll_no*,y=*your_roll_no*+50;
float avg;
avg=(x+y)/2;

Before Executio			After Execution		
n					
x	y	avg	x	y	avg

Lab Session 04

OBJECT

Decision Making Statements(if and if-else, nested structures)

THEORY

Normally, your program flows along line by line in the order in which it appears in your source code. But, it is sometimes required to execute a particular portion of code only if certain condition is true; or false i.e. you have to make decision in your program. There are three major decision making structures. Four decision making structures:

1. *If* statement
2. *If-else* statement
3. *Switch* case
4. Conditional Operator (Rarely used)

The if statement

The **if** statement enables you to test for a condition (such as whether two variables are equal) and branch to different parts of your code, depending on the result. The simplest form of an **if** statement is:

```
if (expression)
    statement;
```

The expression may consist of logical or relational operators like (> >= < <= && ||)

An understanding of if statement is demonstrated with the following example:

```
void main(void)
{
    int var;
    printf("Enter any number;");
    scanf("%d",&var);
    if(var==10)
        printf("The user entered number is Ten");
}
```

The if-else statement

Often your program will want to take one branch if your condition is true, another if it is false.

The keyword **else** can be used to perform this functionality:

```
if (expression)
    statement;
else
    statement;
```

Note: To execute multiple statements when a condition is true or false, parentheses are used.

Consider the following example that checks whether the input character is an upper case or lower case:

```
void main(void)
{
    char ch;
    printf("Enter any character");
    ch=getche();
    if(ch>='A'&&ch<='Z')
        printf("%c is an upper case character",ch);
    else
        printf("%c is a lower case character",ch);
    getch();
}
```

Conditional (Ternary) Operator

The conditional operator (?:) is C's only ternary operator; that is, it is the only operator to take three terms.

The conditional operator takes three expressions and returns a value:

(expression1) ? (expression2) : (expression3);

This line is read as "If expression1 is true, return the value of expression2; otherwise, return the value of expression3." Typically, this value would be assigned to a variable.

An Example:

```
void main(void)
{
    clrscr();
    float per;
    printf("\n Enter your percentage;");
    scanf("%f",&per);
    printf("\n you are");
    printf("%s", per >= 60 ? "Passed": "Failed");
    getch();
}
```

Typecasting

Typecasting allow a variable of one type to act like another for a single operation. In C typecasting is performed by placing, in front of the value, the type name in parentheses.

EXERCISES

2. The programs given below contain some syntax and/or logical error(s). By debugging, mention the error(s) along with their categorization into syntactical or logical error. Also write the correct program statements.

i. // To check whether the number is divisible by 2 or not

```

:
int num;
printf("enter any number") ;
scanf("%f",num);
if(num%2=0)
    printf("Number is divisible by 2");
else
    printf("number is not divisible by 2");
```

—

—

—

—

—

—

ii. // To print your
batch int x
=Your_batch;
if (x==2016)
 printf("Your batch is 2010")
else
 printf("Your batch is %d",x);

Assignments :- Set A

1. Mention the output for the following program :

```
#include<stdio.h>
void main()
{
    int a=100;
    if(a>10)
        printf("Good");
    else if(a>20)
        printf("Better");
    else if(a>30)
        printf("Best");
}
```

2. WAP that takes a number as input from user and checks whether the number is even or odd.

- a) Using if-else
- b) Using conditional operator:

3. WAP to print the ASCII value of any given character

4. WAP to print character value corresponding to any ASCII value

5. WAP that takes a number as input from user and checks whether the number is even or odd

- a) Using if-else
- b) Using conditional operator:

6. WAP that takes three numbers as input from user and finds maximum
 - a) Using if-else
 - b) Using conditional operator:
7. WAP that takes three numbers as input from user and finds minimum
 - a) Using if-else
 - b) Using conditional operator:

Set B

1. WAP to take input student age and check whether given student can vote or not
2. WAP to check whether a given year is leap year or not.
3. WAP to print grade of a student using If Else Ladder Statement
4. WAP to find the roots of quadratic equation.

Lab Assignment 05

OBJECT

Decision Making Statements (Switch Case)

THEORY

The switch Statement

Unlike **if**, which evaluates one value, **switch** statements allow you to branch on any of a number of different values. The general form of the **switch** statement is:

```
switch (expression)
{
    case valueOne: statement;
                    break;
    case valueTwo: statement;
                    break;
    ....
    case valueN:    statement;
                    break;
    default:        statement;
}
```

An Example:

```
void main(void)
{
    clrscr();
    char grade;
    printf("\n Enter your Grade: ");
    grade=getche();
    switch(grade)
    {
        case 'A':
        case 'a':
            printf("\n Your percentage is 80 or above 80 ");
            break;

        case 'B':
        case 'b':
            printf("\n Your percentage is in 70-80 ");
            break;

        default:
            printf("\n Your percentage is below 70 ");

    }
    getch();
}
```


Important Points about Switch Statement

- Switch case performs equality check of the value of expression/variable against the list of case values.
- The expression in switch case must evaluate to return an integer, character or enumerated type.
- You can use any number of case statements within a switch. The expression value is compared with the constant after case.
- The data type of the value of expression/variable must be same as the data type of case constants.
- The break statement is optional. The break statement at the end of each case causes switch statement to exit. If break statement is not used, all statements below that case statement are also executed until it finds a break statement.
- The default code block gets executed when none of the cases matches with the expression. The default case is optional and doesn't require a break statement.
- We don't use those expressions to evaluate switch case, which may return floating point values or strings.

SET A

1. Give the output of following code with explanation

a. `#include<stdio.h>`

`#define L 10`

`void main()`

```
{
    auto a = 10;
    switch (a, a*2)
    {
        case L:
            printf("ABC");
            break;

        case L*2:
            printf("XYZ");
            break;

        case L*3:
            printf("PQR");
            break;

        default:
            printf("MNO");

        case L*4:
            printf("www");
            break;
    }
}
```

```
}
```

b.

```
#include <stdio.h>
int main()
{
    int num = 2;
    switch (num + 2)
    {
        case 1:
            printf("Case 1: ");
        case 2:
            printf("Case 2: ");
        case 3:
            printf("Case 3: ");
        default:
            printf("Default: ");
    }
    return 0;
}
```

2 .WAP to print grade of a student using switch case

3. WAP to accept the week day as number from user and display Monday to Sunday.

4. WAP to check whether a given character is VOWEL or CONSONANT using switch-case

5. WAP to find number of days in a month using switch -case

uses the switch statement to count blanks, tabs, and new-line characters entered from the terminal.

Lab Assignment 06

OBJECT

Study of Loops(While)

THEORY

A loop is used for executing a block of statements repeatedly until a given condition returns false.

Types of Loops

There are three types of Loops:

for Loop

while Loop

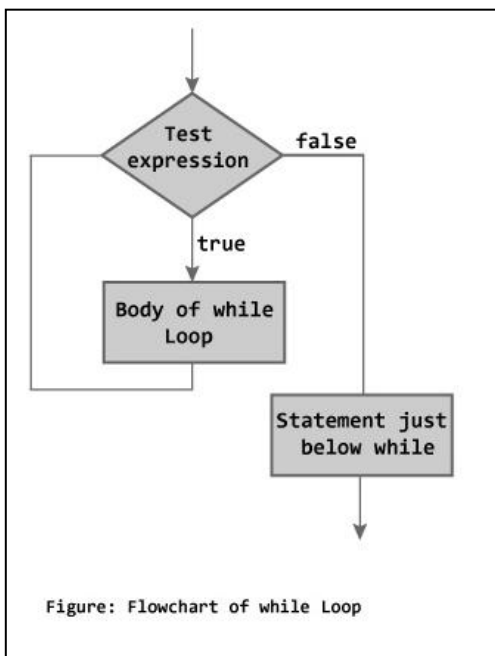
do - while Loop

The while Loop

```
while(condition is true)
{
    Do this;
}
```

This loop runs as long as the condition in the parenthesis is true. Note that there is no semicolon after the **“while”** statement. If there is only one statement in the **“while”** loop then the braces may be removed.

Flowchart:



Example of while loop

```
#include <stdio.h>
int main()
{
    int count=1;
    while (count <= 4)
    {
        printf("%d ", count);
        count++;
    }
    return 0;
}
```

Output:

1 2 3 4

step1: The variable count is initialized with value 1 and then it has been tested for the condition.

step2: If the condition returns true then the statements inside the body of while loop are executed else control comes out of the loop.

step3: The value of count is incremented using ++ operator then it has been tested again for the loop condition.

Examples of infinite while loop

Example 1:

```
#include <stdio.h>
int main()
{
    int var = 6;
    while (var >=5)
    {
        printf("%d", var);
        var++;
    }
    return 0;
}
```

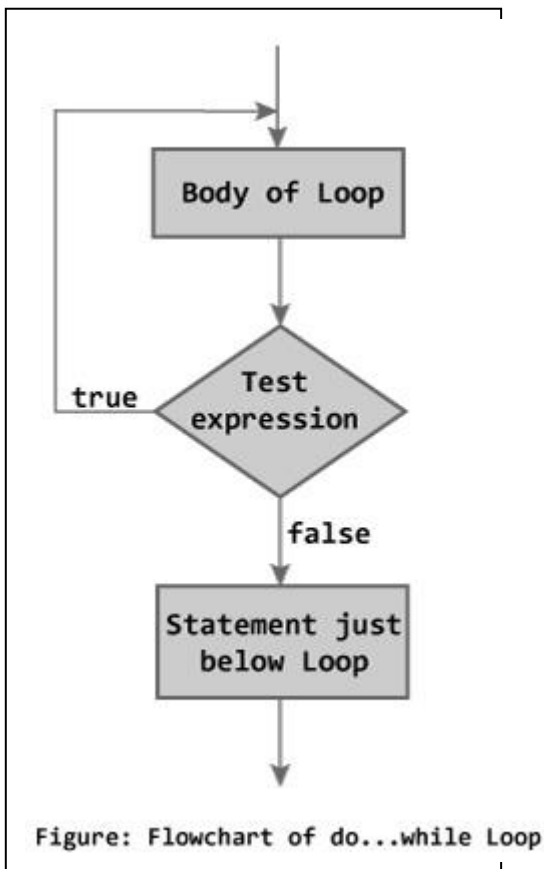
Infinite loop: var will always have value >=5 so the loop would never end.

The do-while Loop

```
do
{
    this;
}
while(condition is true);
```

This loop runs as long as the condition in the parenthesis is true. Note that there is a semicolon after the **“while”** statement. The difference between the **“while”** and the **“do-while”** statements is that in the **“while”** loop the test condition is evaluated before the loop is executed, while in the **“do”** loop the test condition is evaluated after the loop is executed. This implies that statements in a **“do”** loop are executed at least once. However, the statements in the **“while”** loop are not necessarily executed.

Flowchart:



Example of do while loop

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int j=0;
```

```
    do
```

```
    {
```

```

        printf("Value of variable j is: %d\n", j);
        j++;
    }while (j<=3);
    return 0;
}

```

Output:

```

Value of variable j is: 0
Value of variable j is: 1
Value of variable j is: 2
Value of variable j is: 3

```

SET A

1. WAP to add first 10 numbers.
2. WAP to add first n numbers.
3. WAP to add any 10 numbers.
4. WAP to print reverse of any number.
5. WAP to add first 10 numbers.
6. . WAP to find factorial of a given number

SET B

1. WAP to print following pattern.
 - a) for example if n=5 then output should be

```

*
*  *
*  *  *
*  *  *  *
*  *  *  *  *

```

- b) for example if n=5 then output should be

```

          *
        *  *
      *    *  *
    *      *  *  *
  *        *  *  *  *
*          *  *  *  *

```

2. WAP to print the square of number(s) repeatedly till **1** is entered by user. Using ***do-while*** loop.
3. WAP to print following series
 - a) Fibonacci series of n numbers
 - b) $1+3+5+7+\text{-----}+n$

SET C

1. WAP to print following pattern
 - a) for example if $n=5$ then output should be

```
      *
    *   *
  *       *   *
*           *   *   *
*       *       *       *
*   *   *           *   *
```

2. WAP to print table of any given number

Lab Assignment 07

OBJECT

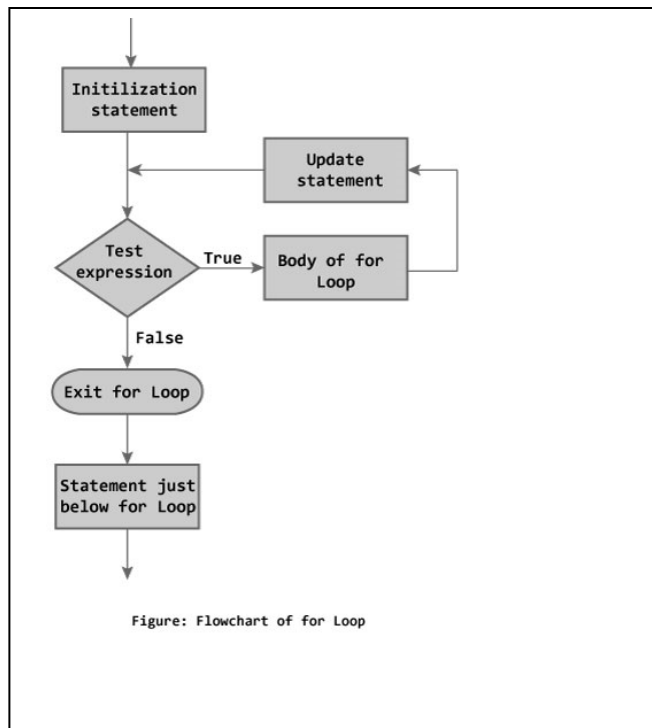
Study of Loops(for loop)

THEORY

For loop
This is one of the most frequently used loop in C programming.

Syntax of for loop:
for (initialization; condition test; increment or decrement)
{
 //Statements to be executed repeatedly
}

Flow Diagram of For loop



Step 1: First initialization happens and the counter variable gets initialized.
Step 2: In the second step the condition is checked, where the counter variable is tested for the given condition, if the condition returns true then the C statements inside the body of for loop gets executed, if the condition returns false then the for loop gets terminated and the control comes out of the loop.
Step 3: After successful execution of statements inside the body of loop, the counter variable is incremented or decremented, depending on the operation (++ or --).

Example of For loop

```
#include <stdio.h>
int main()
{
```



```

int i;
for (i=1; i<=3; i++)
{
    printf("%d\n", i);
}
return 0;
}

```

Output:

```

1
2
3

```

Various forms of for loop in C

1) Here instead of num++, I'm using num=num+1 which is same as num++.

```
for (num=10; num<20; num=num+1)
```

2) Initialization part can be skipped from loop as shown below, the counter variable is declared before the loop.

```
int num=10;
```

```
for (;num<20;num++)
```

Note: Even though we can skip initialization part but semicolon (;) before condition is must, without which you will get compilation error.

3) Like initialization, you can also skip the increment part as we did below. In this case semicolon (;) is must after condition logic. In this case the increment or decrement part is done inside the loop.

```
for (num=10; num<20; )
```

```
{
    //Statements
    num++;
}
```

4) This is also possible. The counter variable is initialized before the loop and incremented inside the loop.

```
int num=10;
```

```
for (;num<20;)
```

```
{
    //Statements
    num++;
}
```

5) As mentioned above, the counter variable can be decremented as well. In the below example the variable gets decremented each time the loop runs until the condition num>10 returns false.

```
for(num=20; num>10; num--)
```

Nested For Loop in C

Nesting of loop is also possible. Lets take an example to understand this:

```
#include <stdio.h>
```

```
int main()
```

```
{
    for (int i=0; i<2; i++)
    {
        for (int j=0; j<4; j++)
        {
            printf("%d, %d\n",i ,j);
        }
    }
    return 0;
}
```

Output:

```
0, 0
```

0, 1
0, 2
0, 3
1, 0
1, 1
1, 2
1, 3

6) Multiple initialization inside for Loop in C

We can have multiple initialization in the for loop as shown below.
for (i=1,j=1;i<10 && j<10; i++, j++)

Exercise as an assignment:-

Solve all programs given for while and do while loops using for loop

SET A

1. WAP to print table of first n numbers in proper format.
2. WAP to check whether a given number is perfect number
3. WAP to check whether a given number is prime number
4. WAP to print first n perfect numbers
5. WAP to print first n prime numbers
6. WAP to check whether a given number is armstrong number or not

SET B

1. WAP a class of n students take an annual examination in 3 subjects. WAP to read the marks obtained by each student in various subjects and to compute and print the total marks obtained by each of them
2. WAP to calculate the series $1+x+x^2+x^3+-----+x^n$
3. WAP to ask the user to input 5 numbers and print out the maximum and minimum numbers from the set.
4. WAP to find power of a number (x^n)
5. WAP to program to count frequency of digits in a given number
6. WAP to program to find HCF(GCD) of two numbers
7. WAP to program to find LCM of two numbers

SET C

1. WAP to find 1s complement of a binary number
2. WAP to convert from Decimal to Binary number system

Lab Assignment 08

OBJECT

Study of Loops (nested loops)

THEORY

C programming allows to use one loop inside another loop.

Syntax

The syntax for a nested for loop statement in C is as follows –

```
for ( init; condition; increment ) {
```

```
    for ( init; condition; increment ) {
        statement(s);
    }
    statement(s);
}
```

The syntax for a nested while loop statement in C programming language is as follows –

```
while(condition) {
```

```
    while(condition) {
        statement(s);
    }
    statement(s);
}
```

The syntax for a nested do...while loop statement in C programming language is as follows –

```
do {
```

```
    statement(s);
```

```
do {
```

```
    statement(s);
```

```
}while( condition );
```

```
}while( condition );
```

A final note on loop nesting is that you can put any type of loop inside any other type of loop. For example, a 'for' loop can be inside a 'while' loop or vice versa.

Example

The following program uses a nested for loop to find the prime numbers from 2 to 100 –

Live Demo

```
#include <stdio.h>
```

```
int main () {
```

```
    /* local variable definition */
    int i, j;
```

```
    for(i = 2; i<100; i++) {
```

```

    for(j = 2; j <= (i/j); j++)
        if(!(i%j)) break; // if factor found, not prime
    if(j > (i/j)) printf("%d is prime\n", i);
}

return 0;
}

```

Output:

```

2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime

```

SET A

1. WAP to print the number pattern.

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

2. WAP to inverted half pyramid using * and numbers

```

* * * * *
* * * *
* * *
* *
*

```

3. WAP to pattern program of stars and alphabets

```
  *  
 *A*  
*A*A*  
*A*A*A*
```

SET B

1. WAP to print all the composite numbers from 2 to a certain number entered by user.(A number is said to be composite if it has at least one factor other than 1 and itself)

2. WAP to print Pascal triangle up to n rows

Lab Assignment 09

OBJECT

Study of Jump statements (exit, goto , continue , break)

Jumping Statements :

C language provides us multiple statements through which we can transfer the control anywhere in the program.

1. break jumping statements.

- By using this jumping statement, we can terminate the further execution of the program and transfer the control to the end of any immediate loop.
- To do all this we have to specify a break jumping statements whenever we want to terminate from the loop.

Syntax: break;

NOTE: This jumping statements always used with the control structure like switch case, while, do while, for loop etc.

NOTE: As break jumping statements ends/terminate loop of one level . so it is refered to use return or goto jumping statements , during more deeply nested loops.

Example: Program based upon break jumping statements:

WAP to display the following output:

1 2 3 4 5

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
for(i=1;i<=10;i++)
{
printf("Enter ",i,"no");
printf(" \t %d ",i);
if(i>5)
break;
}
}
```

2. continue jumping statements.

- By using this jumping statement, we can terminate the further execution of the program and transfer the control to the beginning of any immediate loop.

- To do all this we have to specify a continue jumping statements whenever we want to terminate any particular condition and restart/continue our execution.

Syntax: continue;

NOTE: This jumping statements always used with the control structure like switch case, while, do while, for loop etc.

Example: Program based upon continue jumping statements:

WAP to display the following output:

1 2 3 4 . 6 7 . 9 10

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
for(i=1;i<=10;i++)
{
printf("Enter ",i,"no");
printf(" \t %d ",i);
if(i==5 || i==8)
continue;
}
}
```

3. goto jumping statements.

- By using this jumping statements we can transfer the control from current location to anywhere in the program.
- To do all this we have to specify a label with goto and the control will transfer to the location where the label is specified.

Syntax: goto <label>;

NOTE:

- The control will transfer to those label that are part of particular function, where goto is specified.
- All those labels will not include, that are not the part of a particular function where the goto is specified.

NOTE:

- It is good programming style to use the break, continue and return instead of goto.
- However, the break may execute from single loop and goto executes from more deeper loops.

Example: Program based upon continue jumping statements:

WAP to display the square root of a no, if no. is not positive then re enter the input.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n;
float s;
start:
printf("Enter a no.");
scanf("%d",&n);
s=sqrt(n);
if(n<=0)
goto start;
printf("%f",s);
}
```

Exit

The exit() function:

In C exit() is a predefined/ library function used to terminate the currently running program(process) immediately.

Syntax: void exit(int status);

status -> The status in an integer value returned to the parent process.

Here 0 usually means program completed successfully, and nonzero values are used as error codes. e.g exit(0);

There are also predefined macros EXIT_SUCCESS and EXIT_FAILURE, e.g. exit(EXIT_SUCCESS);

In the C Language, the required header for the exit() function is stdlib.h.

```
#include<stdio.h>
#include<stdlib.h>
```

```
int main () {
    printf("Start of the main()...\n");
    printf("Exiting the main()...\n");
    exit(0);
    printf("End of the program\n");
    return(0);
}
```

Output Start of the main()... Exiting the main()

SET A

Give the output of following codes with proper explanation

```
1. #include <stdio.h>
   int main()
   {
       int a = 0, i = 0, b;
       for (i = 0; i < 5; i++)
       {
           a++;
           continue;
       }
   }
```

```
2. #include <stdio.h>
   int main()
   {
       int a = 0, i = 0, b;
       for (i = 0; i < 5; i++)
       {
           a++;
           if (i == 3)
               break;
       }
   }
```

```
3. #include <stdio.h>
   void main()
   {
       int i = 0, j = 0;
       for (i = 0; i < 5; i++)
       {
           for (j = 0; j < 4; j++)
           {
               if (i > 1)
                   break;
           }
           printf("Hi \n");
       }
   }
```

```
4.
#include <stdio.h>
void main()
{
    int i = 0;
    int j = 0;
    for (i = 0; i < 5; i++)
    {
```

```

        for (j = 0; j < 4; j++)
        {
            if (i > 1)
                continue;
            printf("Hi \n");
        }
    }
}

```

5.

```

#include <stdio.h>
int main()
{
    int num;
    printf("Enter value of num:");
    scanf("%d",&num);
    switch (num)
    {
        case 1:
            printf("You have entered value 1\n");
            break;
        case 2:
            printf("You have entered value 2\n");

        case 3:
            printf("You have entered value 3\n");
            break;
        default:
            printf("Input value is other than 1,2 & 3 ");
    }
    return 0;
}

```

Show the output when num=1 ,num=2,num=4

```

6. #include <stdio.h>
int main()
{
    int var;
    for (var =100; var>=10; var --)
    {
        printf("var: %d\n", var);
        if (var==99)
        {
            break;
        }
    }
    printf("Out of for-loop");
    return 0;
}

```

7.

```
#include <stdio.h>
int main()
{
    int i = 0;
    for (i=0; i<20; i++)
    {
        switch(i)
        {
            case 0:
                i += 5;
            case 1:
                i += 2;
            case 5:
                i += 5;
            default:
                i += 4;
                break;
        }
        printf("%d ", i);
    }
    return 0;
}
```

8.

```
#include <stdio.h>
int main()
{
    int i = 0;
    do
    {
        i++;
        if (i == 2)
            continue;
        printf("In while loop ");
    } while (i < 2);
    printf("%d\n", i);
}
```

9. #include <stdio.h>

```
int main()
{
    int i = 0, j = 0;
    for (i; i < 2; i++){
        for (j = 0; j < 3; j++){
            printf("1\n");
            break;
        }
        printf("2\n");
    }
```

```

    }
    printf("after loop\n");
}

```

10.

```

#include <stdio.h>
int main()
{
    int i = 0;
    char c = 'a';
    while (i < 2){
        i++;
        switch (c) {
            case 'a':
                printf("%c ", c);
                break;
                break;
        }
    }
    printf("after loop\n");
}

```

11.

```

#include <stdio.h>
int main()
{
    printf("%d ", 1);
    goto l1;
    printf("%d ", 2);
l1:goto l2;
    printf("%d ", 3);
l2:printf("%d ", 4);
}

```

12. #include <stdio.h>

```

int main()
{
    printf("%d ", 1);
l1:l2:
    printf("%d ", 2);
    printf("%d\n", 3);
}

```

13.

```

#include <stdio.h>
int main()
{
    printf("%d ", 1);
    goto l1;
}

```

```

    printf("%d ", 2);
}
void foo()
{
    11 : printf("3 ", 3);
}

```

14.

```

#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    while (i < 2)
    {
        11 : i++;
        while (j < 3)
        {
            printf("Loop\n");
            goto 11;
        }
    }
}

```

15.

```

#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    while (11: i < 2)
    {
        i++;
        while (j < 3)
        {
            printf("loop\n");
            goto 11;
        }
    }
}

```

16.

```

#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    11: while (i < 2)
    {
        i++;
        while (j < 3)
        {

```

```

        printf("loop\n");
        goto l1;
    }
}

```

17.

```

#include <stdio.h>
void main()
{
    int i = 0;
    if (i == 0)
    {
        goto label;
    }
    label: printf("Hello");
}

```

18.

```

#include <stdio.h>
void main()
{
    int i = 0, k;
    if (i == 0)
        goto label;
    for (k = 0; k < 3; k++)
    {
        printf("hi\n");
        label: k = printf("%03d", i);
    }
}

```

19.

```

#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("statement-1\n");
    printf("statement-2\n");
    exit(0);
    printf("statement-N\n");

    return 0;
}

```

SET B

1. WAP to calculate the sum of maximum of 10 numbers
// Calculates sum until user enters positive number
2. WAP to calculate sum of maximum of 10 numbers
// Negative numbers are skipped from calculation
3. WAP to print even numbers between 1 to n
4. WAP to print perfect numbers between 1 to n
5. WAP to print accept n characters. Print only vowels .

Lab Assignment 10

OBJECT

Study of Menu driven Programs

To write a menu driven program in C you can do it in two ways.

1. Using an if-else construct.
2. switch statements.

Menu driven program using switch statement is preferred because it is faster, more user-friendly as compare to if-else based menu driven program.

But there are some pitfalls for both the methods:

If you use if-else to make a menu driven program then your program will become hard to read (not user friendly).

And if you use switch statements to make a menu-driven program then your program will be user friendly but the switch keyword is followed by an integer or an expression which evaluates to an integer.

SET A

1. Write a menu-driven program using Switch case to calculate the following:
 1. Area of circle
 2. Area of square
 3. Area of sphere
2. Write a menu driven program to create a simple calculator
(Performs addition, subtraction, multiplication or division depending the input from user)

SET B

1. Write a C program for a menu driven program which has following options:
 1. Factorial of a number.
 2. Prime or not
 3. Odd or even
 4. Exit
2. Write a menu driven C program using switch-case to find:
 - (a) Sum of the digits of number
 - (b) product of the digits of number.

SET C

1. Write a menu driven C program using switch-case to perform following operations
 - A. bitwise and (&)
 - B. bitwise or(|)

Lab Session 11

OBJECT

Study of Functions (User defined)

THEORY

A function is a block of code that performs a particular task.

There are many situations where we might need to write same line of code for more than once in a program. This may lead to unnecessary repetition of code, bugs and even becomes boring for the programmer. So, C language provides an approach in which you can declare and define a group of statements once in the form of a function and it can be called and used whenever required.

These functions defined by the user are also known as User-defined Functions

C functions can be classified into two categories,
Library functions
User-defined functions

User-defined functions

There are four types of functions depending on the return type and arguments:

1. Functions that take nothing as argument and return nothing.
2. Functions that take arguments but return nothing.
3. Functions that do not take arguments but return something.
4. Functions that take arguments and return something.

General syntax for function declaration is,

returntype functionName(type1 parameter1, type2 parameter2,...);

Like any variable or an array, a function must also be declared before its used. Function declaration informs the compiler about the function name, parameters is accept, and its return type. The actual body of the function can be defined separately. It's also called as Function Prototyping. Function declaration consists of 4 parts.

returntype

function name

parameter list

terminating semicolon

returntype

When a function is declared to perform some sort of calculation or any operation and is expected to provide with some result at the end, in such cases, a return statement is added at the end of function body. Return type specifies the type of value(int, float, char, double) that function is expected to return to the program which called the function.

Note: In case your function doesn't return any value, the return type would be void.

functionName

Function name is an identifier and it specifies the name of the function. The function name is any valid C identifier and therefore must follow the same naming rules like other variables in C language.

parameter list

The parameter list declares the type and number of arguments that the function expects when it is called. Also, the parameters in the parameter list receives the argument values when the function is called. They are often referred as formal parameters.

Function definition Syntax

Just like in the example above, the general syntax of function definition is,

```
returntype functionName(type1 parameter1, type2 parameter2,...)
{
    // function body goes here
}
```

The first line returntype functionName(type1 parameter1, type2 parameter2,...) is known as function header and the statement(s) within curly braces is called function body.

Note: While defining a function, there is no semicolon(;) after the parenthesis in the function header, unlike while declaring the function or calling the function.

functionbody

The function body contains the declarations and the statements(algorithm) necessary for performing the required task. The body is enclosed within curly braces { ... } and consists of three parts.

local variable declaration(if required).

function statements to perform the task inside the function.

a return statement to return the result evaluated by the function(if return type is void, then no return statement is required).

Calling a function

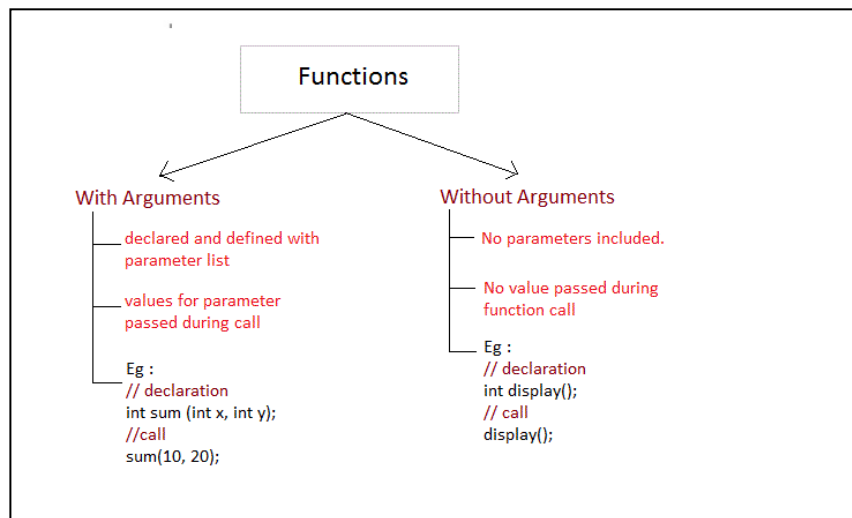
When a function is called, control of the program gets transferred to the function.

`functionName(argument1, argument2,...);`

In the example above, the statement `multiply(i, j);` inside the `main()` function is function call.

Passing Arguments to a function

Arguments are the values specified during the function call, for which the formal parameters are declared while defining the function.



Returning a value from function

A function may or may not return a result. But if it does, we must use the return statement to output the result. return statement also ends the function execution, hence it must be the last statement of any function. If you write any statement after the return statement, it won't be executed. The datatype of the value returned using the return statement should be same as the return type mentioned at function declaration and definition. If any of it mismatches, you will get compilation error.

Consider a simple example of function declaration, definition and call.

```
void function1(void);  
void function2(void)
```

```

    {
        printf("Writing in Function2\n");
    }
void main(void)
{
    printf("Writing in main\n");
    function1( );
}
void function1(void)
{
    printf("Writing in Function1\n");
    function2( );
}

```

Consider another example that adds two numbers using a function **sum()** .

```

void sum(void);
void main(void)
{
    printf("\nProgram to print sum of two numbers\n");
    sum(void);
}
void sum(void)

{
    int num1,num2,sum;
    printf("Enter 1st number:");
    scanf("%d",&num1);
    printf("Enter 2nd number:");
    scanf("%d",&num2);
    sum=num1+num2;
    printf("Sum of %d+%d=%d",num1,num2,sum);
}

```

Solve Set A ,B and C using user defined functions

SET A

1. Identify the errors (if any) in the following code:

a) func(int a,int b)

```

{
int a; a=20; return a;
}

```

b) #include<stdio.h> int main()

```

{

```

```

int myfunc(int); int b; b=myfunc(20); printf("%d",b); return 0;

}
int myfunc(int a)
{
a > 20? return(10): return(20);
}

```

2. WAP to calculate cube of a number
3. WAP to print your name 10 times. The function can take no arguments and should not return any value.
4. WAP to print Table of an Integer Number.
5. WAP to find SUM and AVERAGE of two integer Numbers .

SET B

- 1.WAP to print the GCD of 4 given numbers using function to calculate the GCD of two numbers*/
2. WAP to find the factorial of an Integer
3. WAP to Calculate the value of nCr
5. main() is a function. Write a function which calls main(). What is the output of this program?

SET C

- 1 . WAP that takes a number as input and print its binary equivalent.

Lab Assignment 12 and Assignment 13

Study of Functions

call By value

Function call by value is the default way of calling a function in C programming

Important terms:-

What is Function Call By value?

When we pass the actual parameters while calling a function then this is known as function call by value. In this case the values of actual parameters are copied to the formal parameters. Thus operations performed on the formal parameters don't reflect in the actual parameters.

Actual parameters: The parameters that appear in function calls.

Formal parameters: The parameters that appear in function declarations.

call By Reference

Unlike call by value, in this method, address of actual arguments (or parameters) is passed to the formal parameters, which means any operation performed on formal parameters affects the value of actual parameters.

SET A

Trace the output .Give proper explanation

```
1.  # include<stdio.h>
    void fun(int*, int*);
    int main()
    {
        int i=5, j=2;
        fun(&i, &j);
        printf("%d, %d", i, j);
        return 0;
    }
    void fun(int *i, int *j)
    {
        *i = *i**i;
        *j = *j**j;
    }
```

1. Using a function, swap the values of two variables. The function takes two values of Variables as arguments and returns the swapped values.(call by value)

2. Write function definition that takes two complex numbers as argument and prints their sum.

Write a C program to print all even or odd numbers in given range.

3. check if an integer (entered by the user) can be expressed as the sum of two prime numbers of all possible combinations with the use of functions.

Lab Session 14

OBJECT

Study of Recursive Functions

Recursion

Recursion is an ability of a function to call itself.

An example: A program that calculates the following series using recursion.

$$n + (n-1) + (n-2) + + 3 + 2 + 1$$

```
int add(int);
void main(void)
{
    int num,ans;
    printf("Enter any number:");
    scanf("%d",&num);
    ans=add(num);
    printf("Answer=%d",ans);
    getch();
}
int add(int n)
{
    int result;
    if(n==1)
        return 1;
    result=add(n-1) + n;
    return result;
}
```

SET A

1. Trace the output

```
#include<stdio.h>
int i;
int fun();

int main()
{
    while(i)
    {
        fun();
        main();
    }
}
```



```

    }
    printf("Hello\n");
    return 0;
}
int fun()
{
    printf("Hi");
}

```

1. WAP to Find the Sum of Natural Numbers using Recursion

2. WAP to find factorial of a number using Recursion

3. WAP to Find the GCD using Recursion

SET B

4. WAP to Find the power of a number using Recursion

5. WAP to find the factorial of an Integer (using recursion)

6. WAP to print Fibonacci series of n numbers (using recursion)

7. WAP that takes a number as input and print its binary equivalent.(using recursion)

8. Write a C program to find reverse of any number using recursion.

Lab Session 15

OBJECT

Study of Arrays

THEORY

An array is a collection of data storage locations, each of which holds the same type of data. Each storage location is called an element of the array. You declare an array by writing the type, followed by the array name and the subscript. The subscript is the number of elements in the array, surrounded by square brackets.

For example , `int LongArray[25];`

declares an array of 25 integers, named LongArray. When the compiler sees this declaration, it sets aside enough memory to hold all 25 elements. Because each long integer requires 2 bytes, this declaration sets aside 50 contiguous bytes of memory.

Elements of an Array and How to access them?

You can access elements of an array by indices.

Suppose you declared an array mark as above. The first element is mark[0], second element is mark[1] and so on.

Few key notes:

Arrays have 0 as the first index not 1. In this example, mark[0]

If the size of an array is n, to access the last element, (n-1) index is used. In this example, mark[4]

Suppose the starting address of mark[0] is 2120d. Then, the next address, a[1], will be 2124d, address of a[2] will be 2128d and so on. It's because the size of a float is 4 bytes.

How to initialize an array in C programming?

It's possible to initialize an array during declaration. For example,
`int mark[5] = {19, 10, 8, 17, 9};`

Another method to initialize array during declaration:

```
int mark[] = {19, 10, 8, 17, 9};
```

How to insert and print array elements

Consider the following program that take 10 numbers as input in an array and then print that array.

```
#include<stdio.h>
```

```

void main(void)
{
    clrscr();
    int arr[10];
    for(int i=0;i<10;i++)
    {
        printf("\n\tEnter element %d:",i+1);
        scanf("%d",&arr[i]);
    }
    clrscr();
    for(int j=0;j<10;j++)
    printf("\n\n\tElement %d is %d",j+1,arr[j]);
    getch();
}

```

EXERCISES

SET A

- 1. WAP to Calculate Sum & Average of an Array of size 20**
- 2. WAP to Find the Largest Number in an Array**
- 3. WAP to Find the Largest Two Numbers in a given Array**
- 4. WAP to Put Even & Odd Elements of an Array in 2 Separate Arrays**
- 5. WAP to Insert an Element in a Specified Position in a given Array**

SET B Program to be done using function

- 1. WAP to Sort the Array in an Ascending Order**
- 2. WAP Input an Array, Store the Squares of these Elements in an Array & Print it.**
- 3. WAP Increment every Element of the Array by one & Print Incremented Array.**
- 4. WAP to find 2 Elements in the Array such that Difference between them is Largest**

SET C

- 1. WAP to Find the Second Largest & Smallest Elements in an Array**
- 2. WAP to Delete the Specified Integer from an Array(using function)**

Lab Session 16

OBJECT

Study of Multidimensional Arrays

In C programming, you can create an array of arrays known as multidimensional array. For example,

```
float x[3][4];
```

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as table with 3 row and each row has 4 column.

Similarly, you can declare a three-dimensional (3d) array. For example,

```
float y[2][4][3];
```

Here, The array y can hold 24 elements.

You can think this example as: Each 2 elements have 4 elements, which makes 8 elements and each 8 elements can have 3 elements. Hence, the total number of elements is 24.

How to initialize a multidimensional array?

There is more than one way to initialize a multidimensional array.

Initialization of a two dimensional array

// Different ways to initialize two dimensional array

```
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};
```

```
int c[][3] = {{1, 3, 0}, {-1, 5, 9}};
```

```
int c[2][3] = {1, 3, 0, -1, 5, 9};
```

Above code are three different ways to initialize a two dimensional arrays.

Initialization of a three dimensional array.

You can initialize a three dimensional array in a similar way like a two dimensional array. Here's an example,

```
int test[2][3][4] = {  
    { {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },  
    { {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }  
};
```

SET A

1. WAP to store temperature of two cities for a week and display it.
2. WAP to find the sum of two matrices of order 2×2 using multidimensional arrays.
3. WAP to store values entered by the user in a three-dimensional array and display it.

SET B

1. WAP to multiply two matrices of size m and n.
2. WAP to find the transpose of a matrix
3. WAP find addition of any given row number.

SET C

1. WAP to add the diagonal elements
2. WAP to find the sum all the elements

Lab Session 17

OBJECT

Study of Standard Library functions

The standard functions are built-in functions. In C programming language, the standard functions are declared in header files and defined in .dll files. In simple words, the standard functions can be defined as "the ready made functions defined by the system to make coding more easy". The standard functions are also called as library functions or pre-defined functions. In C when we use standard functions, we must include the respective header file using #include statement. For example, the function printf() is defined in header file stdio.h (Standard Input Output header file). When we use printf() in our program, we must include stdio.h header file using #include<stdio.h> statement.

Few examples of header files with standard functions provided by C Programming Language are

Header File	Purpose	Example Functions
stdio.h	Provides functions to perform standard I/O operations	printf(), scanf()
conio.h	Provides functions to perform console I/O operations	clrscr(), getch()
math.h	Provides functions to perform mathematical operations	sqrt(), pow()
string.h	Provides functions to handle string data values	strlen(), strcpy()
stdlib.h	Provides functions to perform general functions	calloc(), malloc()
time.h	Provides functions to perform operations on time and date	time(), localtime()
ctype.h	Provides functions to perform - testing and mapping of character data values	isalpha(), islower()

SET A

1. WAP to check whether given character is
 - a. Alphabet or not
 - b. Digit or not
 - c. symbol or not
2. WAP to check whether a given character is upper case. If it is upper case convert it to lower case

SET B

1. WAP to Find Square root
2. WAP to Find power of x^y

SET C

1. Check Armstrong Number of n digits



Savitribai Phule Pune University, Pune

F. Y. B. C. A. (Science) Semester-I

Lab Course – II

Fundamentals of Computer

Workbook

Name: _____

College Name: _____

Roll No.: _____ Division: _____

Academic Year: _____

Editors:

Introduction

1. About the Workbook:

This workbook is intended to be used by FYBCA (Science) students for the Programming in C& Fundamental of computers Assignments in Semester-I. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus.

2. The objectives of this Workbook are:

- 1) Defining the scope of the course.
- 2) To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
- 3) To have continuous assessment of the course and students.
- 4) Providing ready reference for the students during practical implementation.
- 5) Provide more options to students so that they can have good practice before facing the examination.
- 6) Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. How to use this Workbook:

The workbook is divided into two sections. Section-I is related to Programming in C assignments and Section-II is related to Fundamentals of computer assignments.

The Section-I (Programming in C) is divided into thirteen assignments. Each assignment has several SET. It is mandatory for students to complete all the SET in given slot.

The Section-II (Fundamentals of computer) is divided into Fourteen assignments. The assignments comprise of activities to be carried out on given databases. The students have to create database, insert appropriate records and then perform the activities specified in each of the assignments. A pool of databases will get created as student progresses through the assignments and these databases can be repeatedly used in subsequent assignments.

Each FOC assignment has several SET. It is mandatory for students to complete all the SETs in given slot.

4. Instructions to the students:

Please read the following instructions carefully and follow them.

- Students are expected to carry this workbook every time they come to the lab for practical.
- Students should prepare for the assignment by reading the relevant material which is mentioned in ready reference.
- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments if needed.
- Students will be assessed for each assignment on a scale from 0 to 5

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

5. Instruction to the Instructors:

- Make sure that students should follow above instructions.
- Explain the assignment and related concepts using white board if required or by demonstrating the software.
- Give specific input to fill the blanks in queries which can vary from student to student.
- Evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- The value should also be entered on assignment completion page of the respective Lab course.

6. Instructions to the Lab administrator:

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side areas given below:

- Server and Client Side-(Operating System) Fedora Core Linux/Windows
- Turbo C

Table of Contents

Section I: Operating System

Assignment 1	09
---------------------------	-----------

DOS Commands

Assignment 2	16
---------------------------	-----------

Operating System Installation (Demo)

Section II: Microsoft Office

Assignment 3	23
---------------------------	-----------

Microsoft word I

Assignment 4	28
---------------------------	-----------

Microsoft word II

Assignment 5	29
---------------------------	-----------

Spreadsheet I

Assignment 6	33
---------------------------	-----------

Spreadsheet II

Assignment 7	35
---------------------------	-----------

Microsoft Power point I

Assignment 8	39
---------------------------	-----------

Microsoft Power point II

Section III: Shell Commands

Assignment 9	41
---------------------------	-----------

Shell Commands

Section IV : HTML

Assignment 10.....47

Introduction to HTML

Assignment 11.....54

Working with List and Hyperlink

Assignment 12.....58

Working with Table

Assignment 13.....62

Working with Frames

Assignment 14.....66

Working with Forms

Assignment Completion Sheet

Lab Course II			
Fundamentals of Computers			
Sr. No.	Assignment Name	Marks (out of 5)	Instructor Sign
1	DOS Commands		
2	Operating System Installation		
3	Microsoft Word I		
4	Microsoft Word II		
5	Spreadsheet I		
6	Spreadsheet II		
7	Microsoft PowerPoint I		
8	Microsoft PowerPoint II		
9	Shell Command		
10	Introduction to HTML		
11	Working with List and Hyperlink		
12	Working with Table		
13	Working with Frames		
14	Working with Forms		
Total (Out of 70)			
Total (Out of 30)			

This is to certify that Mr./Ms. _____
Has successfully completed the Fundamental of Computers course work for Lab
Course II and has scored _____ Marks out of 30.

Instructor

H.O. D / Coordinator

Internal Examiner

External Examiner

Section I: Operating System

Assignment No 1

DOS COMMANDS

Syntax Notes

To be functional, each DOS command must be entered in a particular way: this command entry structure is known as the command's "syntax." The syntax "notation" is a way to reproduce the command syntax in print.

For example, you can determine the items that are optional, by looking for information that is printed inside square brackets. The notation [d:], for example, indicates an optional drive designation. The command syntax, on the other hand, is how YOU enter the command to make it work.

Command Syntax Elements

1. Command Name

The DOS command name is the name you enter to start the DOS program (a few of the DOS commands can be entered using shortcut names). The DOS command name is always entered first. In this book, the command is usually printed in uppercase letters, but you can enter command names as either lowercase or uppercase or a mix of both.

2. Space

Always leave a space after the command name.

3. Drive Designation

The drive designation (abbreviated here as "d:") is an option for many DOS commands. However, some commands are not related to disk drives and therefore do not require a drive designation. Whenever you enter a DOS command that deals with disk drives and you are already working in the drive in question, you do not have to enter the drive designator. For example, if you are working in drive A (when the DOS prompt A> is showing at the left side of the screen) and you want to use the DIR command to display a directory listing of that same drive, you do not have to enter the drive designation. If you do not enter a drive designation, DOS always assumes you are referring to the drive you are currently working in (sometimes called the "default" drive).

4. A Colon

When referring to a drive in a DOS command, you must always follow the drive designator with a colon (:) (this is how DOS recognizes it as a drive designation).

5. Pathname

A pathname (path) refers to the path you want DOS to follow in order to act on the DOS command. As described in Chapter 3, it indicates the path from the current directory or subdirectory to the files that are to be acted upon.

6. Filename

A filename is the name of a file stored on disk. As described in Chapter 1, a filename can be of eight or fewer letters or other legal characters.

7. Filename Extension

A filename extension can follow the filename to further identify it. The extension follows a period and can be of three or fewer characters. A filename extension is not required.

8. Switches

Characters shown in a command syntax that are represented by a letter or number and preceded by a forward slash (for example, "/P") are command options (sometimes known as "switches"). Use of these options activate special operations as part of a DOS command's functions.

9. Brackets

Items enclosed in square brackets are optional; in other words, the command will work in its basic form without entering the information contained inside the brackets.

10. Ellipses

Ellipses (...) indicate that an item in a command syntax can be repeated as many times as needed.

11. Vertical Bar

When items are separated by a vertical bar (|), it means that you enter one of the separated items. For example: ON | OFF means that you can enter either ON or OFF, but not both.

DOS Commands are divided into 2 types:

1. Internal Commands

These are for performing basic operations on files and directories and they do not need any external file support.

2. External Commands

These external commands are for performing advanced tasks and they do need some external file support as they are not stored in COMMAND.COM

In MS-DOS, keyboard shortcuts involving handy ones like Functional keys, arrows, pipe character (" | "), asterisk (*), ?, [] and ESC are of great help for recalling to searching to clearing command line etc., Here are few of them:

- UP (↑) and DOWN (↓) arrows recall previously entered commands.
- ESC clears the present command line. It abandons the currently construct command and the next prompt appears.
- F1 retypes one character at a time from the last command entry from the current cursor position.
- F2 retypes all characters from the last command entry up to the one identical to your next keystroke. It asks you to enter char to copy up to and retypes the last command up to that char.
- F3 retypes all remaining characters from the last command entry.
- F4 stores all characters beginning at the first match with your next keystroke and ending with the last command entry.
- F5 or F8 keys give all the previously typed commands.
- F6 places a special end-of-file code at the end of the currently open file. Sometimes referred to as Ctrl+z or ^z.
- F7 key displays command history and ALT+F7/ESC hides it.
- F9 is used to select a command by number. Just enter the command number and it fetches the command line for you.
- Pipe character (" | ") combines several series of commands or programs inter-dependent.
- Name enclosed within [] indicate a sub-directory.
- Asterisk (*) is used to represent zero or more any characters.
- ? is used to present zero or single character.

MS-DOS commands perform tasks like:

- Manage files and directories
- Maintain Disks
- Configure Hardware and Networking
- Optimize the use of memory
- Customize MS-DOS

Commonly used Internal DOS Commands

1. Date

This command is used to display the system current date setting and prompt you to enter a new date.

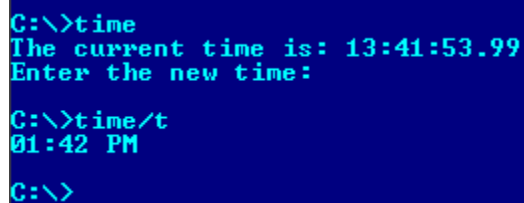
The syntax is: **DATE [/T | date]**

If you type DATE without parameters then it displays current date and prompts to enter new date. We should give new date in mm-dd-yy format. If you want to keep the same date just Press ENTER. DATE command with /T switch tells the command to just output the current system date, without prompting for a new date.

2. TIME

This command is used to displays or set the system time.

The syntax is: **TIME [/T | time]**



```
C:\>time
The current time is: 13:41:53.99
Enter the new time:

C:\>time/t
01:42 PM

C:\>
```

Same as DATE command, typing TIME with no parameters displays the current time and a prompt for a new one. Press ENTER to keep the same time. TIME command used with /T switch tells the command to just output the current system time, without prompting for a new time.

3.COPY CON

It is used to create a file in the existing directory. Here CON is a DOS reserved word which stands for console.

Syntax is: **COPY CON filename** after that press Enter and start typing your text and after you're done typing your text, to save and exit hit F6 key.

4. TYPE

This command is used to display the contents of a text file or files. The syntax is: **TYPE**
[drive:][path]filename

Now, let's try to display the contents of the file na

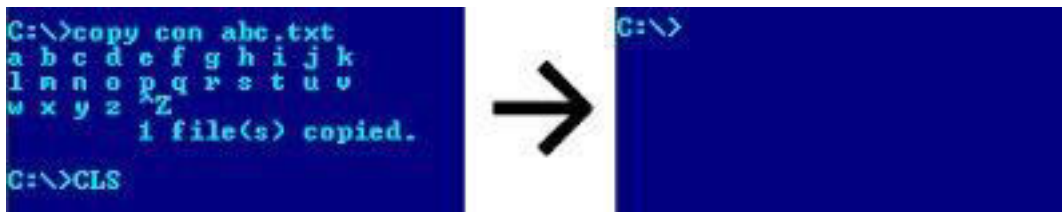
```
C:\>copy con filename
-----
COPY CON Command is used to
create a file in the existing
directory, so here we're creating
a file named filename in C drive.

To save and exit type F6, which I'm
gonna do now ^Z
      1 file(s) copied.

C:\>
```

5. CLS

It is used to clear the screen. Syntax is **CLS**



6. REN

This command is used to change/modify the name of a file or files.

Syntax is: **REN [drive:] [path] filename1 filename2.**

Here, filename1 is source file for which you wanted to change the name, and filename2 will obviously become your new file name. Also note that you cannot specify a new drive or path for your destination file.

7. DIR

This command displays a list of files and subdirectories in a directory. Syntax is: **DIR**

[drive:] [path] [filename] [/A[:attributes]] [/B] [/C] [/D] [/L]

[/N] [/O[:sortorder]] [/P] [/Q] [/S] [/T[:timefield]] [/W] [/X] [/4]

Here,

[drive:][path][filename] Specifies drive, directory, and/or files to list.

/A:attributes

Displays files with specified attributes. The possible attributes are as follow: D **I** Directories, R **I** Read-only files, H **I** Hidden files, A **I** Files ready for archiving, S **I** System files, – Prefix meaning not

/B

display in bare format with no heading information or summary

Using this attribute with dir by default displays the thousandseparator in file sizes. To disable display or separator use /-C

/C

/D

Displays file list sorted by column.

/L

Uses lowercase in listing file names and sub-directories.

/N

Display in new long list format where filenames are on the far right.

/O:sortorder

Displays list by files in sorted order. The sortorder attributes are as follow: N **I** By name (alphabetic), S **I** By size (smallest first), E **I** By extension (alphabetic), D **I** By date/time (oldest first), G **I** Group directories first, – Prefix to reverse order

/P

/P

Display page wise pausing after each screenful of information and prompts to press any key to continue.

/Q

Displays the owner of a file or files.

Displays files in specified directory and all subdirectories. Bear caution in using this in your root directory as you may end up in overflowing information. To stop the screen overflow at any point hit Pause-Break key.

/S

8. PATH

This command displays the path that how we have come to the present position or sets a search path for executable files.

Its Syntax is **PATH** **[[drive:]path[;...];;%PATH%]**

Typing PATH without any parameters displays the current path under current directory. Typing PATH ; clears all search-path settings and direct cmd.exe to search only in the current directory. And including %PATH% in the new path setting causes the old path to be appended to the new setting.

9. VER

This command displays the version of the Microsoft Windows running on your computer.

10. VOL

It displays the disk volume label and serial number, if they exist for the drive specified. If no drive is specified it displays for the active drive.

Syntax is **VOL** **[drive:]**

```
C:\>vol
Volume in drive C has no label.
Volume Serial Number is EC21-77CD

C:\>vol e:
Volume in drive E is New Volume
Volume Serial Number is 60B4-4F09

C:\>
```

11. DEL/ERASE

Used to delete one or more files.

Syntax is **DEL** **[/P] [/F] [/S] [/Q] [/A[:attributes]] names**

12. COPY

This command is useful in copying one or more files to another file or location. Syntax is **COPY** **[/D] [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] source [/A | /B] [+ source [/A | /B] [+ ...]] [destination [/A | /B]]**

The different switches that can be used with this command as follow along with their use.

13. MD, CD and RD

- a. **MD (or MKDIR)** command stand for make directory and it is used to create a directory.
Syntax is **MD** **[drive:]path**
- b. **CD (or CHDIR)** stands for create or change directory and it allows to display the name of or change the current directory or rather we can say come out of a directory. Syntax is **CD** **[/D] [drive:]path**
1 Typing **CD drive:** displays the current directory in the specified drive. This CD (or CHDIR)

command does not treat spaces as delimiters due to which it allows to CD into a subdirectory name that contains a space without surrounding the name with quotes.

For example:

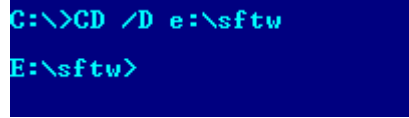
CHDIR program filesmozilla firefox is

the same as:

CHDIR "program filesmozilla firefox"

❶ If you type *CD* without any parameters it displays current drive and directory. *CD..* specifies that you want to change to the higher directory in the current path. Whereas, using *CD* you can directly change to parent/root directory from any location in the current drive.

❷ Using */D* switch changes current drive in addition to current directory for a drive.



```
C:\>CD /D e:\sftw
E:\sftw>
```

- c. **RD (or RMDIR)** command removes or deletes a directory. There are two conditions to remove any directory – (1) Directory to be removed should be empty. and (2) We should be outside the directory we are commanding to delete.

Syntax is **RD [/S] [/Q] [drive:]path**

Here, using the switch */S* removes a directory tree meaning it removes all directories and files in the specified directory in addition to the directory itself. And using */Q* is the quiet mode that doesn't ask for ok approval to remove a directory tree.

Assignment 2

Operating System Installation (Demo)

Windows 7 Installation Overview

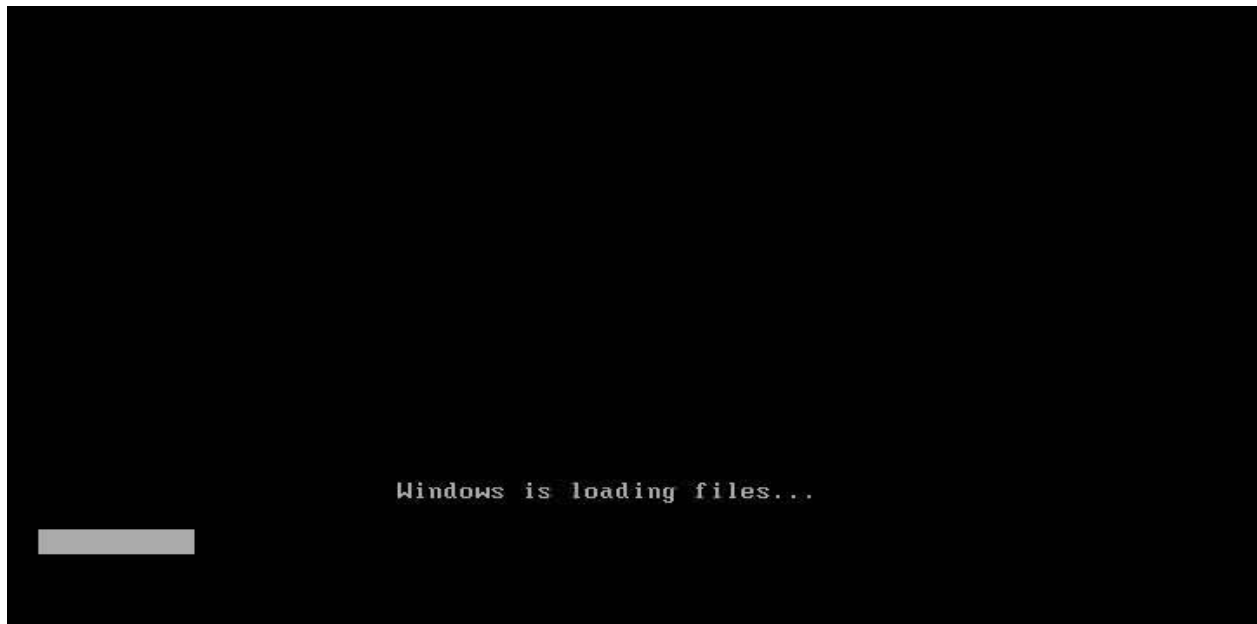
You should have to get your System Specifications and then search for it on Google. If Windows 7 Supports your system then begin this tutorial step by step. In this tutorial you will learn How To Install Windows 7.

Things You'll Need Before Windows 7 Installation

If you already have boot- able CD/DVD then you can skip this part. Otherwise if you need to have Windows 7 ISO download then [click here](#). Also if you want to install windows 7 with USB then make flash drive boot- able by following this method.

How To Install Windows 7 – Steps

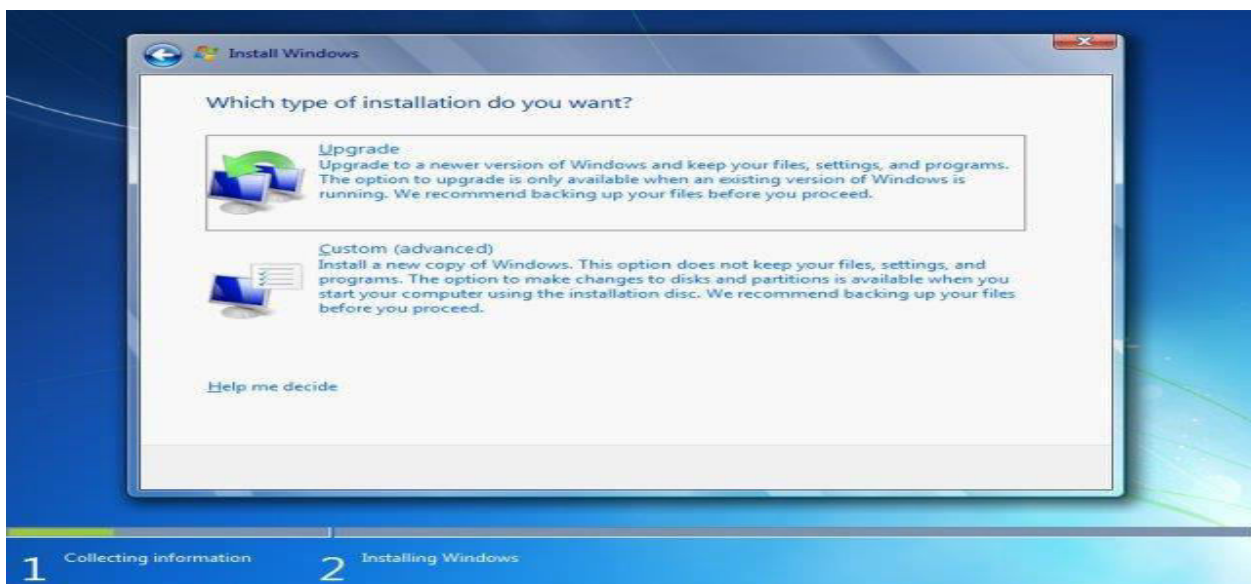
Step#1 Turn ON your PC and Press 'F2' Continuously. There will come up and option to boot through CD/DVD. Select that option. Windows will start loading its files.



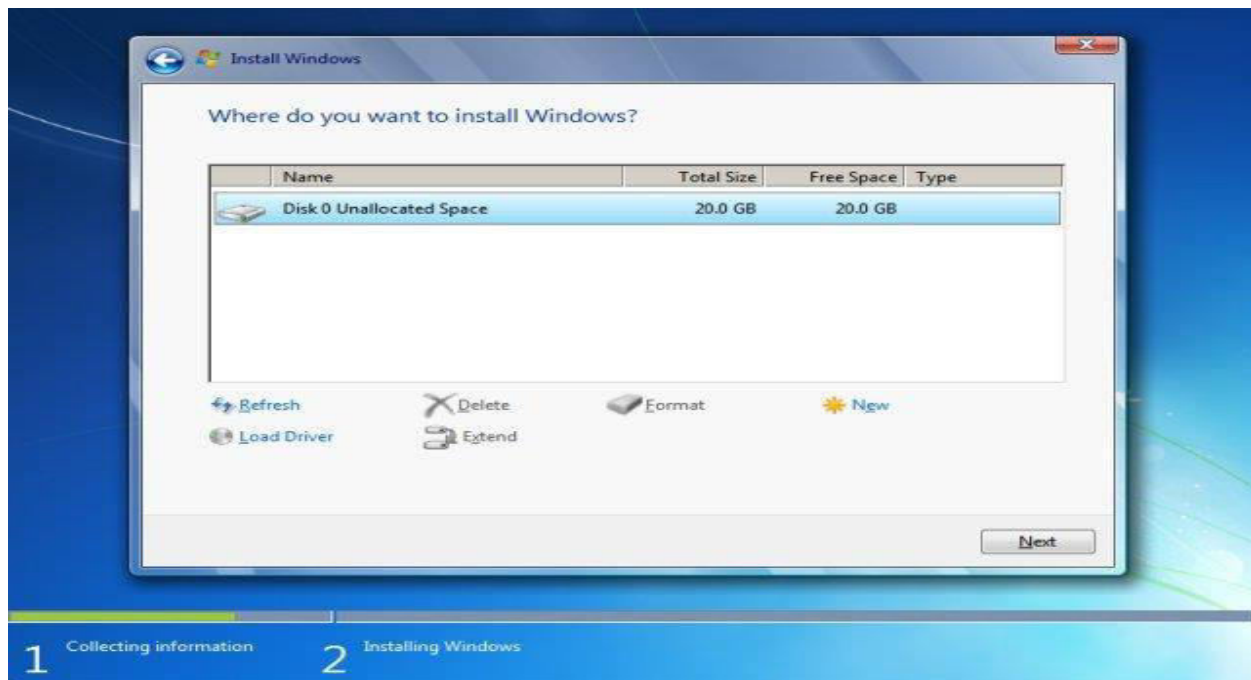
Step#2 Now you will get the Windows Setup Window. This is the part to select Language for your windows. Select 'English' and click Next. Also there will be a 'INSTALL NOW' button. Click on it and proceed to next step.



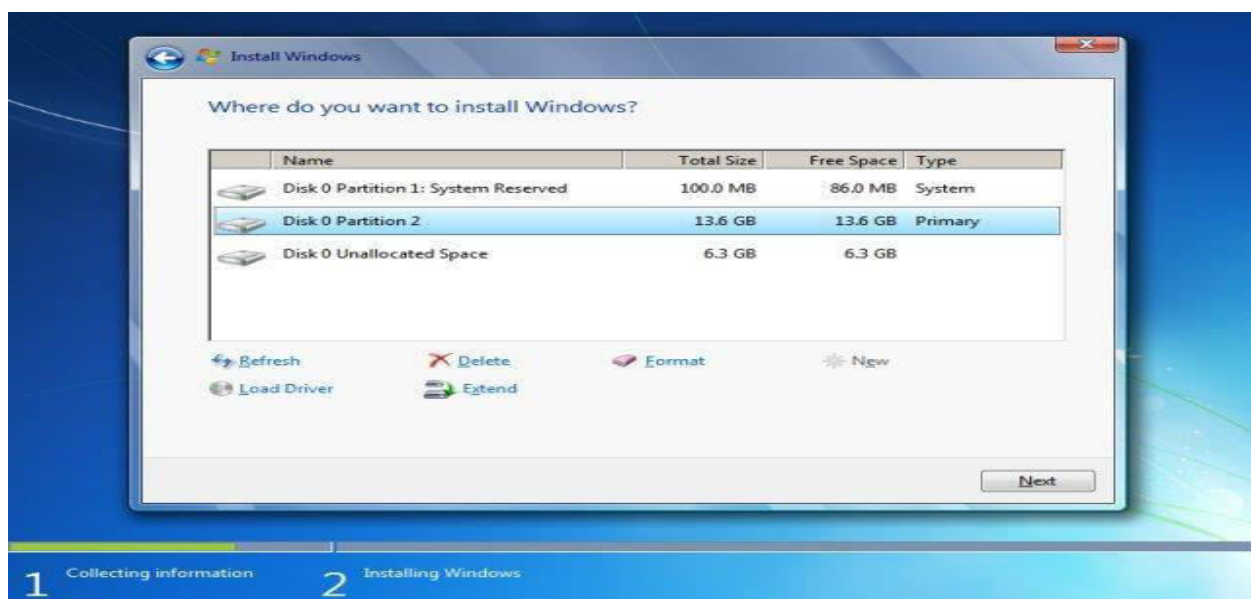
Step #3



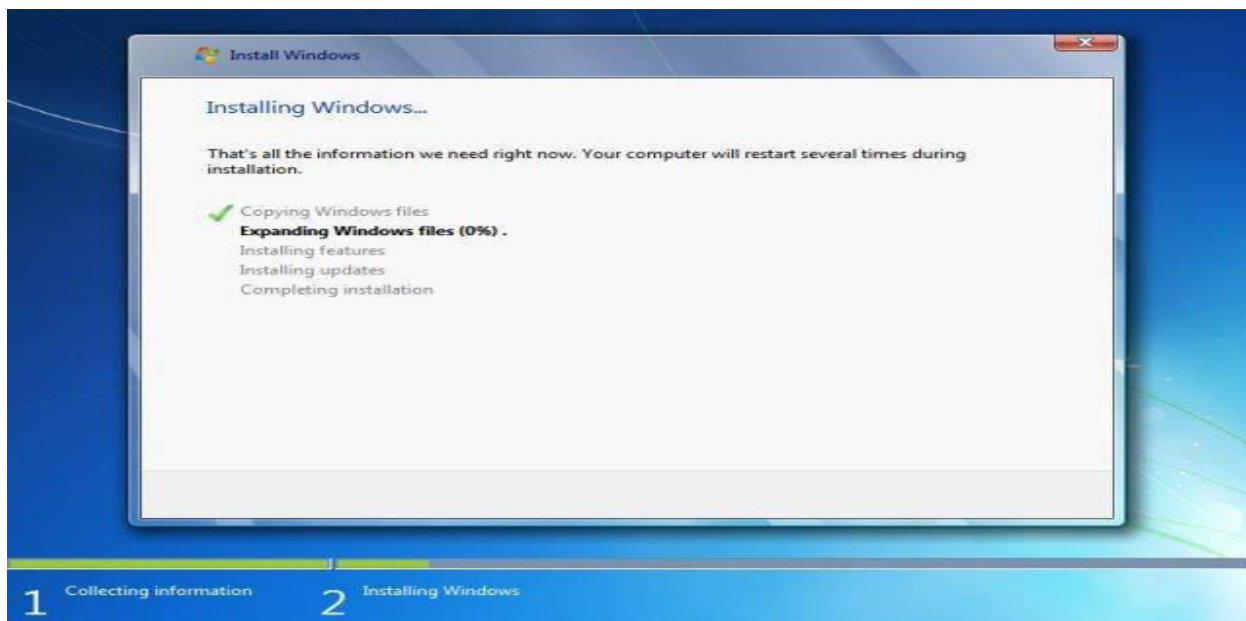
Step#4 In this step you will do partitioning of your drive. Be careful, this is the most important part of the Installation. In this you will allocate spaces to your drive. If you want to create a new drive, simply click on a drive and then click 'NEW'. A new drive will be created.



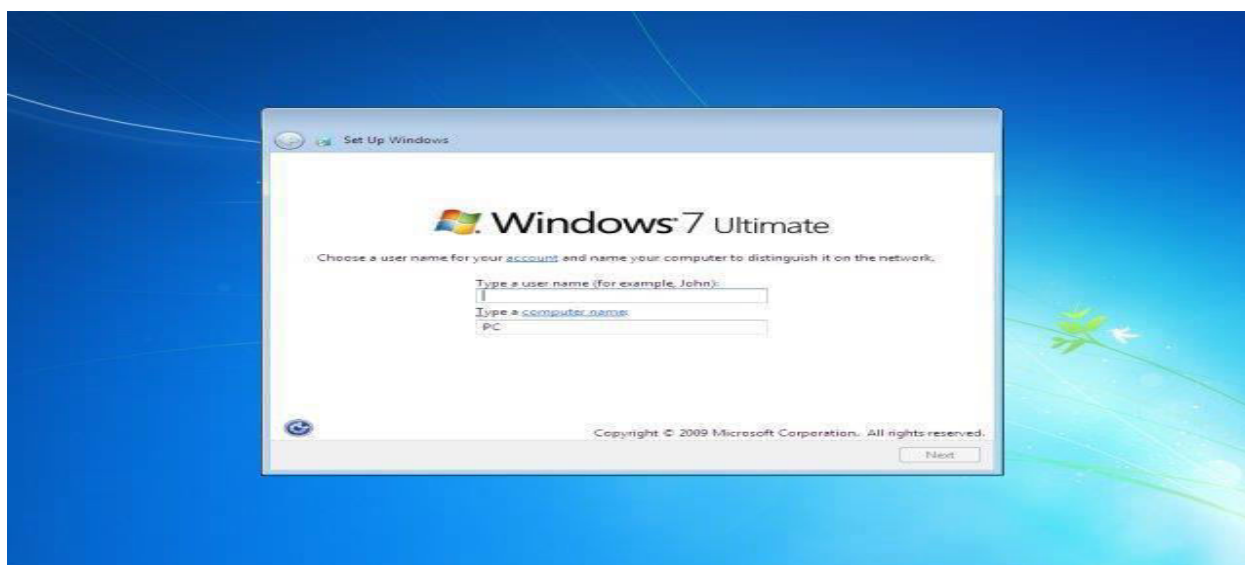
Step #5



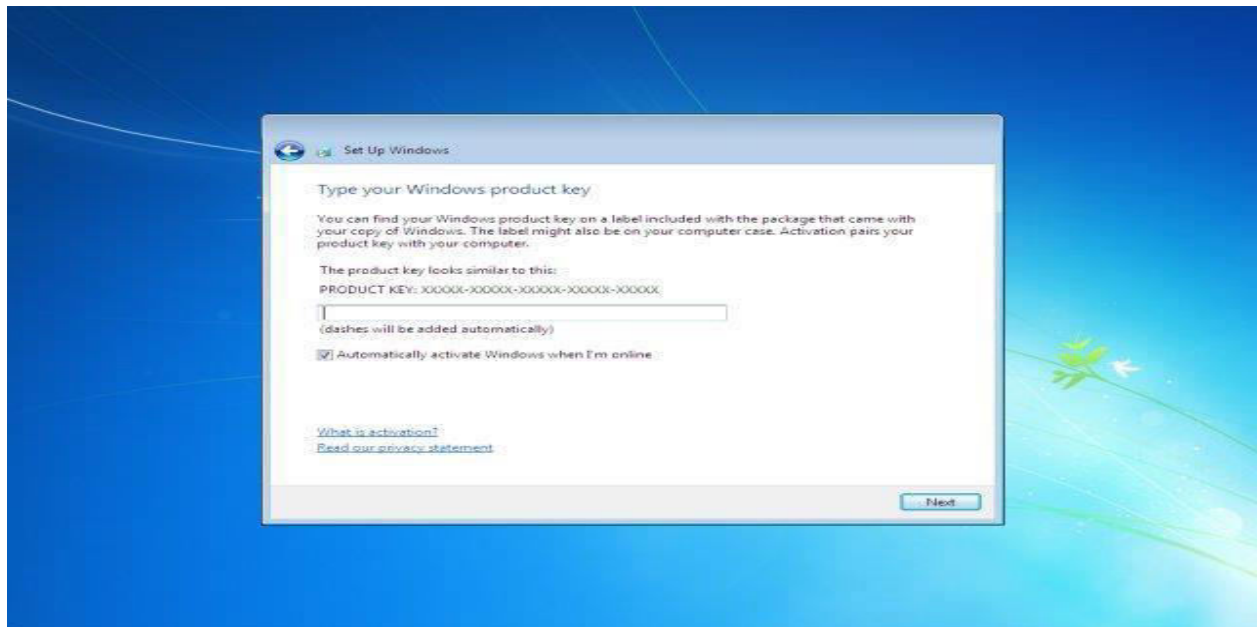
Step#6 Now you windows will start installing its files. Grab a cup of coffee and wait for a few minutes while it install. During this process don't plug in or off your device. It might cause interruption and you might loose your data and have to begin the process all over again.



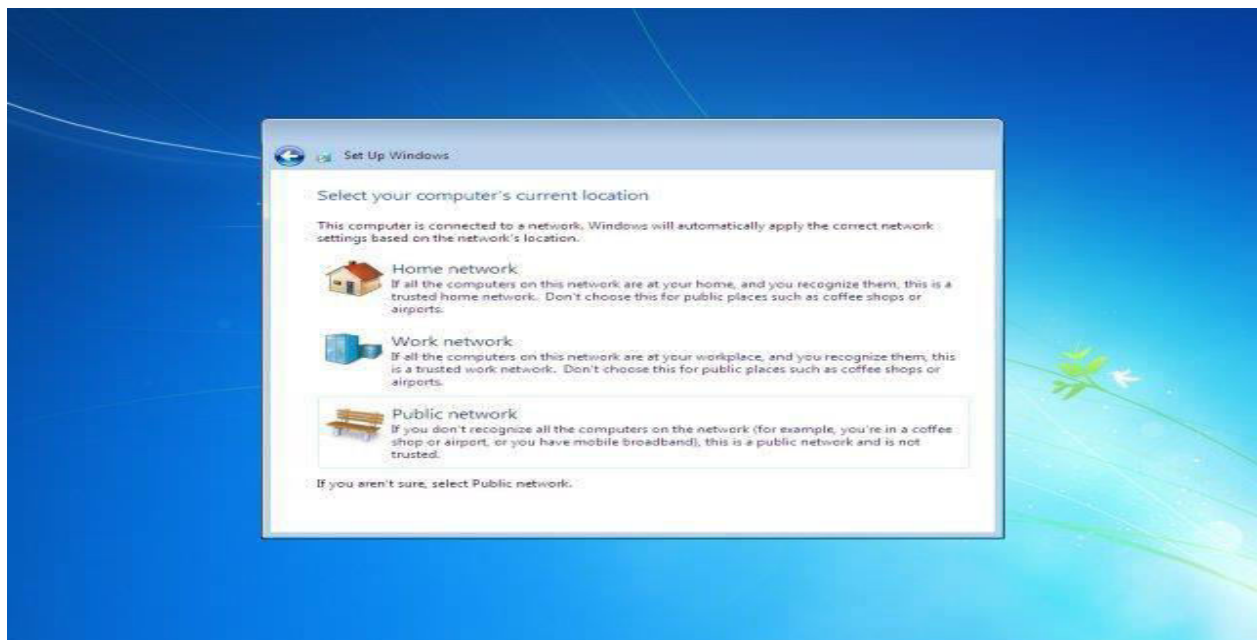
Step #7



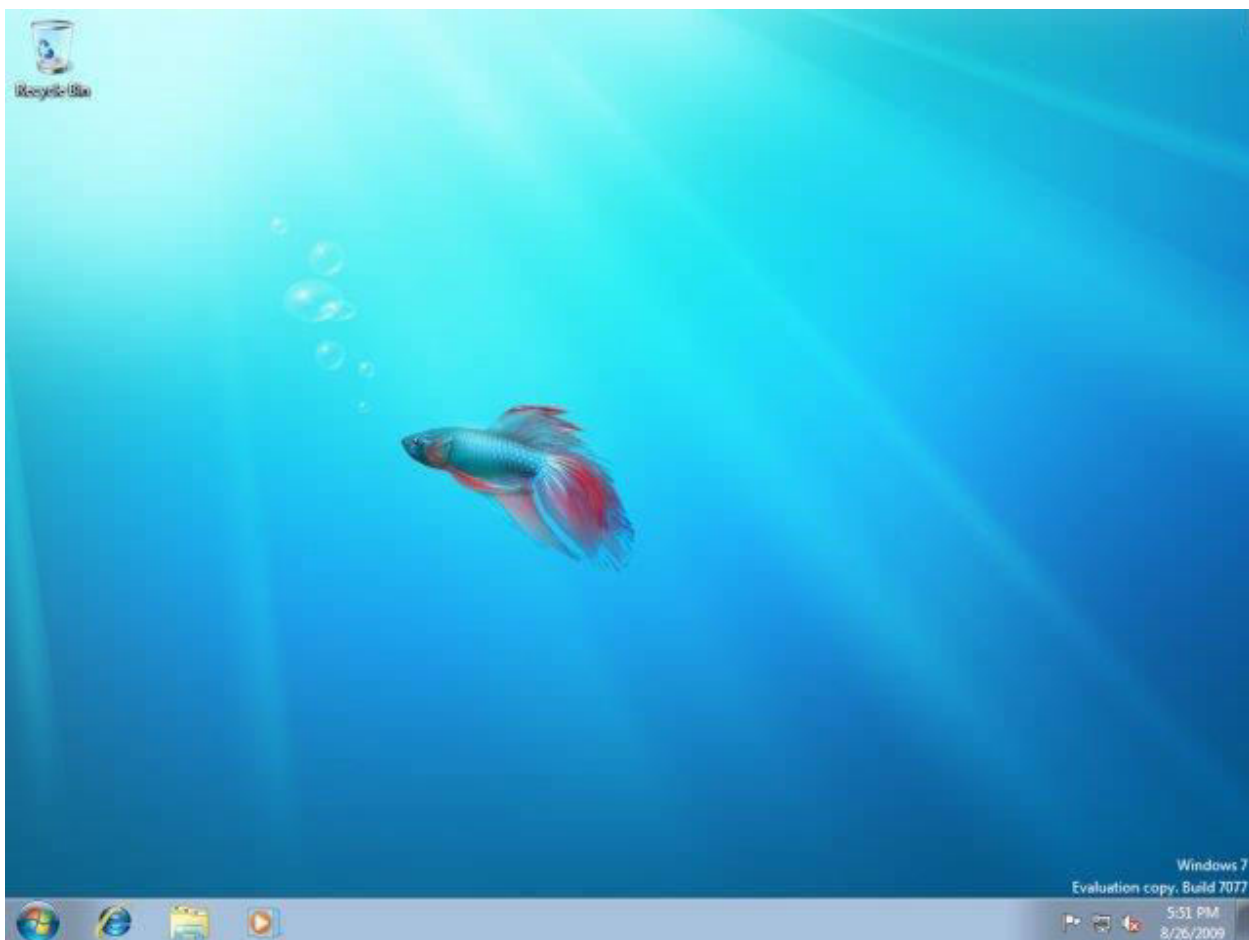
Step#8 In this step you have to activate your windows. Simply look at the back of your Windows CD/DVD cover there will be a PRODUCT KEY. Add this key into your PC and Click 'NEXT'.



Step #9



Last Step – Congratulations:- You have installed you windows. Now you can see is your desktop. It is simple to use, setup your desktop and enjoy!



Section II: Microsoft Office

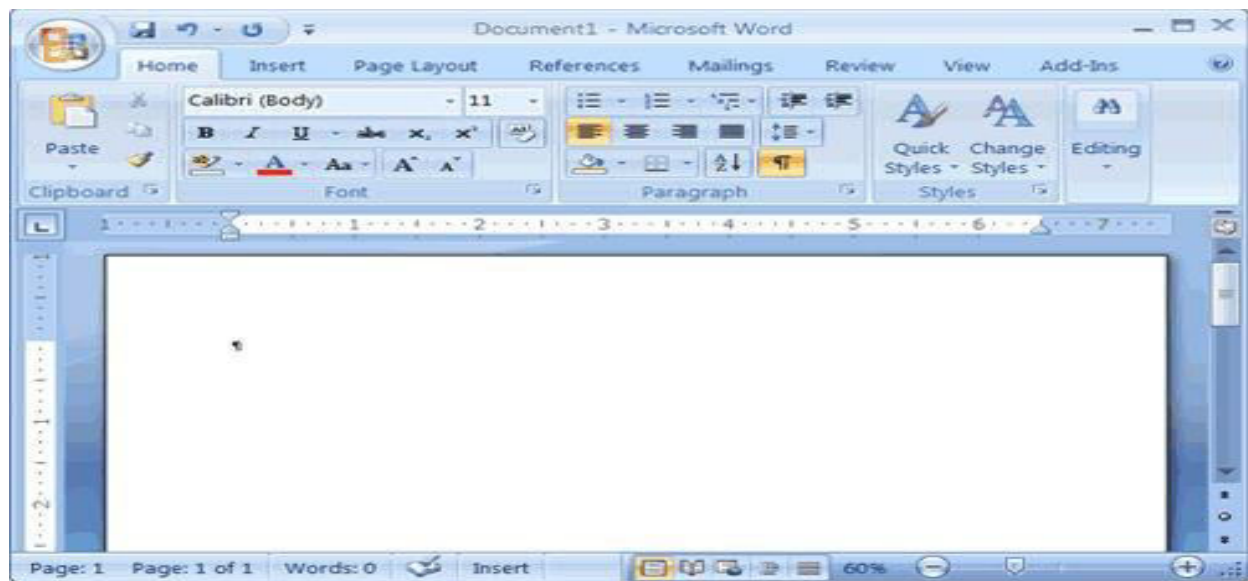
Assignment No 3: INTRODUCTION TO MS-OFFICE

Microsoft office is a set of inter related desk top applications ,servers and services, collectively refers to as an office suit for the micro soft windows and MAC OSX operating systems .



MS WORD:

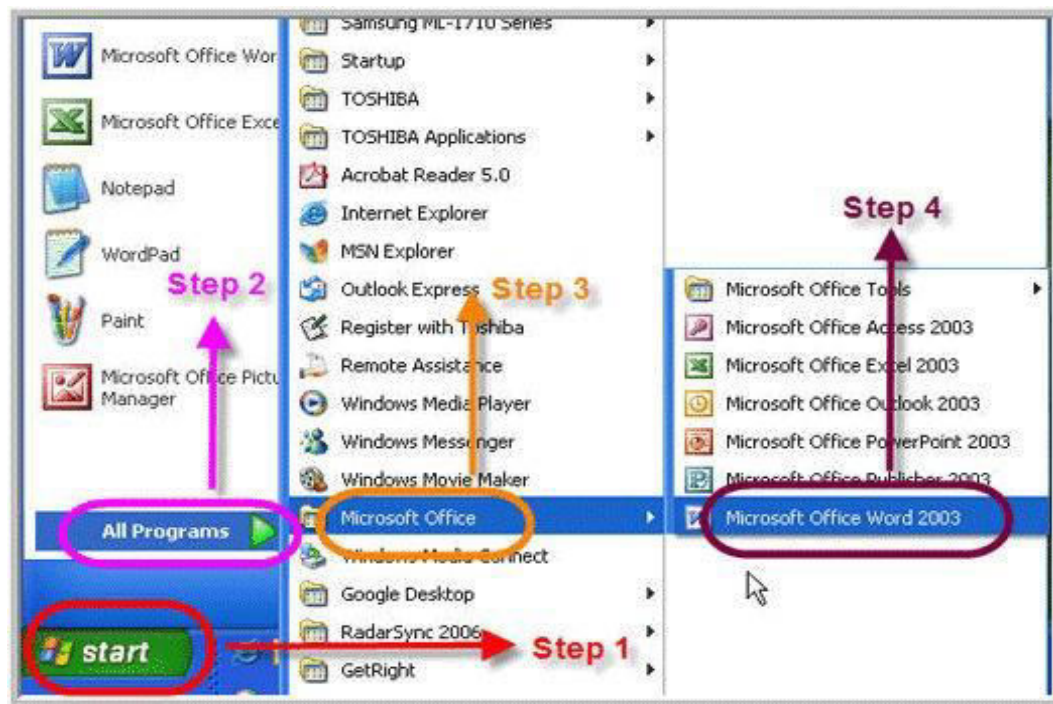
Microsoft Word is a word processing software package. we can use it to type letters, reports, and other documents. In Word 2007, how a window displays depends on the size of your window, the size of Your monitor and the resolution to which your monitor is set. Resolution determines how much information your computer monitor can display.



STARTING MS WORD:-

Two ways of starting MSWORD:-

Double click on Microsoft word icon on the desk top. Click on start ->programs->ms office ->msword.



The Microsoft Office Button

In the upper-left corner of the Word 2007 window is the Microsoft Office button. When you click the button, a menu appears. You can use the menu to create a new file, open an existing file, save a file, and perform many other tasks.



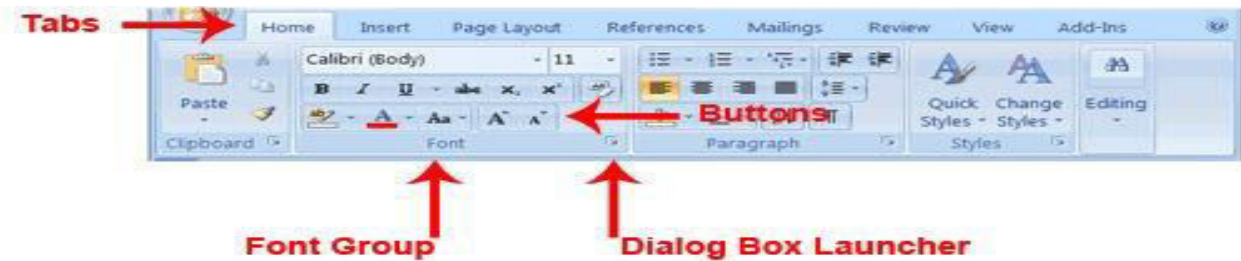
The Quick Access Toolbar

The Quick Access toolbar provides you with access to commands you frequently use. By default Save, Undo, and Redo appear on the Quick Access toolbar. You can use Save your file, Undo to rollback an action you have taken, and Redo to reapply an action you have rolled back.



The Ribbon

We use the Ribbon to issue commands. The Ribbon is located near the top of the screen, below the Quick Access toolbar. At the top of the Ribbon are several tabs; clicking a tab displays several related command groups. Within each group are related command buttons. You click buttons to issue commands or to access menus and dialog boxes



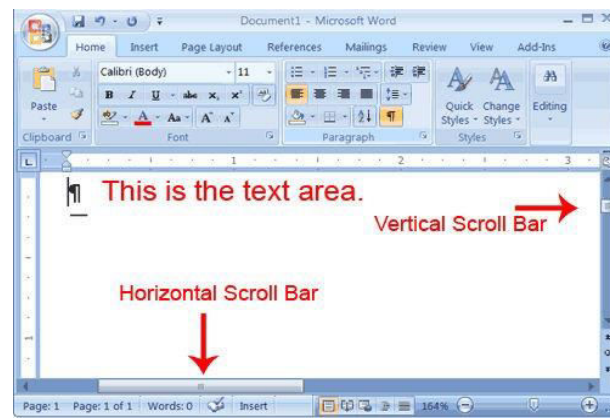
The Ruler

We can use the ruler to change the format of your document quickly



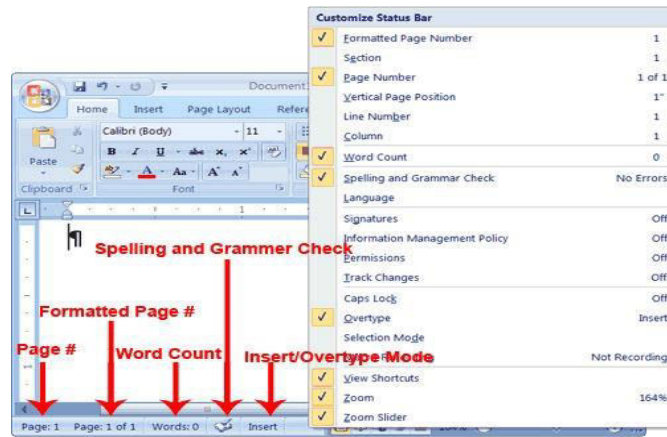
The Text Area

You type your document in the text area. The blinking vertical line in the upper-left corner of the text area is the cursor.



The Status Bar

The Status bar appears at the very bottom of your window and provides such information as the current page and the number of words in your document.



Assignment

SET A

1. Create a document and
 - a. Put Bullets and Numbers
 - b. Apply various Font parameters.
 - c. Apply Left, Right, and Centre alignments.
 - d. Apply hyperlinks
 - e. Insert pictures
 - f. Insert ClipArt
 - g. Show the use of WordArt
 - h. Add Borders and Shading
 - i. Show the use of Find and Replace.
 - j. Apply header/footers
2. Create a student table and do the following:
 - a. Insert new row and fill data
 - b. Delete any existing row
 - c. Resize rows and columns
 - d. Apply border and shading
 - e. Apply merging/splitting of cells
 - f. Apply sort
 - g. Apply various arithmetic and logical formulas.

SET B

1. Create a document to show the use of Watermark.
2. Create a document with at least three paragraphs and perform editing operations.

SET C

Create a formal letter using a suitable word processing package, like MS Word, to place a purchase order for procurement of books. Use tables for list of books.

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment no. 04

SET A

1. Create a document with at least two pages to show use of header and footer.
2. Create a document to add various shapes with color and text options. Add border to this pages.

SET B

1. Using word, create September month timetable. It should include the following
 - a. time slot
 - b. days of week
 - c. border
 - d. subject in each slot
 - e. proper heading
 - f. footer: 'FY BCA TIME TABLE'
2. On Microsoft Word, write a leave application to your College Principal, asking for 3 days holiday, as you have to attend your sister's wedding at Nagpur. Create a table for 3 days function, you will be attending. You will be marked on font, font size, letter format, tabbing, line spacing & table.

SET C

1. Create a formal letter using MS Word, to place a purchase order for procurement of books. Use tables for list of books.
2. Open a new document & save it as '3G_Hours_Firstname_Lastname'
 - a. Insert a table that is 3 columns wide by 4 rows high.
 - b. Enter the information in the table as shown below.

Type a.m. and p.m. exactly as shown.

Monday-Thursday	1 p.m.	10 p.m.
Friday	1 p.m.	11 p.m.
Saturday and holidays	11 a.m.	11 p.m.
Sunday	NOON	11 p.m.

- c. . Select the table and change the settings to **AutoFit to Contents**.
- d . **Align left** the **first column** of the table.

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

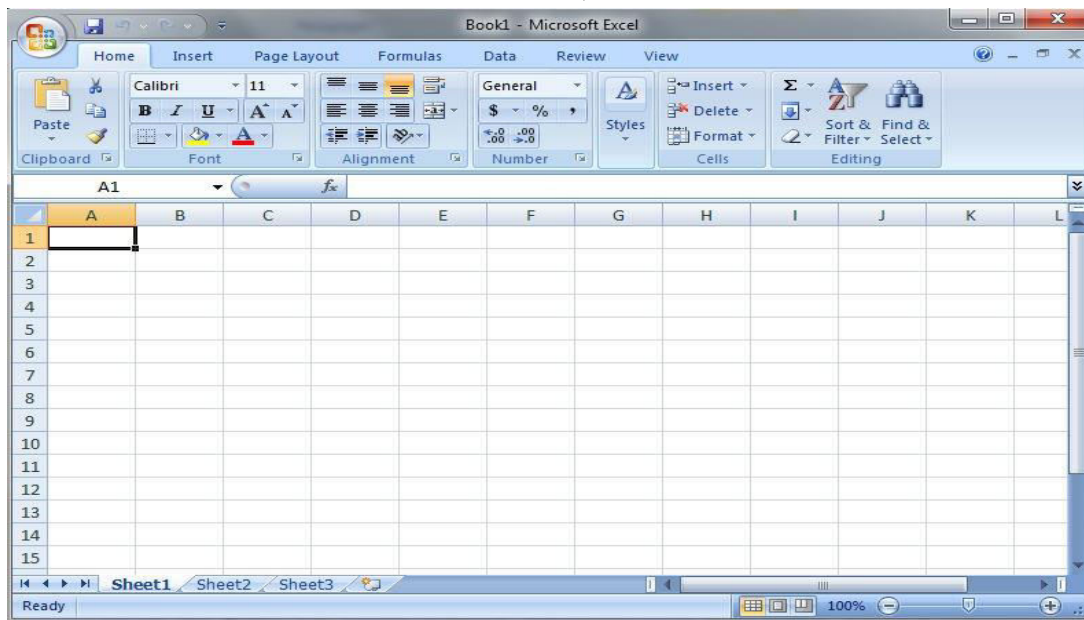
Signature of Instructor

Assignment No 5

TO STUDY MICROSOFT EXCEL

WHAT IS MICROSOFT EXCEL?

Ms-excel is a windows based application package, which is also the member of ms-office family. It can be used to automate accounting, scientific calculation related tasks such as calculations and analysis of data. Ms-excel is easily customizable. It provides a very comfortable environment and assists the user in several ways. When excel starts, worksheet opens automatically. The major elements of the excel screens are toolbars, worksheet and status bar.



➤ What is menu bar?

Menu bar contains several menus which can be invoked by simply clicking on them by a mouse or by using the short cut key combinations from the keyboard .These menus provide access to different commands of excel. Menu bar displays the list of all these menu groups. The menus in excel are

Name	shortcut key
------	--------------

File	alt +f (related to file/folder)
------	---------------------------------

Edit	alt +e (related to word processing & text editing)
------	--

View alt +v (related to page setting & Layout) Insert

alt +i (related to insertion of types of items)

Format alt +o (for text formatting of Cells, rows)

Tools alt +t (contains tools like auto & spell check)

Data alt +d (has data processing commands)

➤ Important Excel functions

Function	What it does	Examples
sum (n1, n2)	Calculates the total of all values in Parenthesis	=sum(s2;s3) displays the total of all the numbers
Average (n1,n2)	Calculates the avg. Of a group of values	=avg.(t2:t5) display avg. of all no. In cell t2
Count (v1,v2)	Counts the no. Of cells that contain numeric values	=count(w1,w2) if every cell in this range contains
Max (n1,n2)	Finds highest and lowest value in the list	=max(a1,a2)displays highe
Today (n1,n2)	Displays today's date in the cell	=today () calculate the no. Of days
Percentage	Calculates percentage of a group of values	=c2*100/d2 displays the percentage of all the no.

➤ What is a workbook and worksheet?

A workbook is a multi page excel document. Each page in the workbook is called a work sheet, and the active worksheet is displayed in the document window. At the left end of the horizontal scroll bar are sheet tabs and navigation buttons .Use the sheet tabs to move to another worksheet and the navigation buttons to scroll through the sheet tabs.

➤ What are worksheet components?

Each worksheet is divided into columns, rows and cells separated by gridlines. The first column a, and the letter A appear in the column heading. The horizontal rows are numbered. Each worksheet has 256 columns (a through iv) and 65536 rows.

➤ Insert a worksheet?

To insert a worksheet, go to insert menu and choose worksheet

➤ **Delete a worksheet?**

To delete a worksheet, click on the work sheet name tab, go to edit menu and choose delete worksheet.

➤ **Creating graphs and charts**

Excel has powerful graphics and charting features. These are very useful in presentation, in decision making and in analyzing the data.

1. Open the salary worksheet.
2. Select the cells a8 till a13.
3. Hold down ctrl and select the cells h8 till h13.
4. Now we have two ranges of cells, which are required for the pie chart- the names and the net pay of the employees.
5. Click on the chart wizard on the formatting toolbar. The chart wizard appears.
6. In the chart wizard, under the standard types tab, choose pie as chart type.
7. In the sub-type section select the second figure-pie with a 3-d visual effect.
8. Click next. The next step of the chart wizard appears.
9. Click the finish button. The chart appears as an object in the salary worksheet.
10. Click the save button on the standard toolbar to save the worksheet and the chart.

Assignment

SET A

1. Create a table for student information with at least 7 columns.
2. Create table and perform all mathematical functions.

SET B

1. Open a new Workbook and perform these operations:

a. Enter this data in Sheet1 of the workbook

Name	Maths	English	Computers
Neha	94	80	96
Ankit	90	70	89
Pooja	76	78	70
Rahul	80	77	89
Mayank	78	76	87

- b. Rename worksheet as 'Merit List'
- c. Insert one more worksheet in your workbook & rename it to 'Marks'
- d. Save the workbook by the name 'Report Card'.

2. Create a list of your friends in class using the Custom List option. Enter the names of those friends in an Excel worksheet using the fill handle.

SET C

1. Generation of Electricity Bill.
2. Generation of Salary statement of an employee

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment No: 06

SET A

1. Create a worksheet to compute mean/median/mode of student percentage.
2. Generate graph to show the production of goods in a company during the last five years.
3. Generation of Telephone Bill

SET B

1. The following are the salaries of five employees. Create a File in MS EXCEL

Pay Roll No	Name	Salary Rs.	Part time Rs.	Accounts
1011	Prasanna	10000	900	1800
1012	Anitha	14000	800	1600
1013	Ravi	18000	700	1700
1014	Saritha	15000	600	1600
1015	Mallika	17000	500	1800

Using Conditional Formatting list out employees who got

- a) Less than Rs. 15000 as salary
 - b) More than Rs. 700 as Part time
 - c) Between Rs. 1600 and Rs. 1800 as Arrears.
2. Create a MS-Excel worksheet Display a Pie Chart for following data, Also calculate total marks and average marks using functions.

Roll No	Marks out of 500
1	432
2	300
3	400
4	302
5	455

SET C

1. Generate the following worksheet

Roll No.	Marks
2050	67
2051	49
2052	40
2053	74
2054	61
2055	57
2056	45

and do the following:

- a. Create chart of the marks.
 - b. Compute sum of marks using autosum, autocalculate and sum function.
 - c. Compute average of marks.
 - d. Show pass or fail if marks are above 50 or less than 50
 - e. Put header and footer in the spread sheet.
2. Create a Spreadsheet in MS-EXCEL and enter the marks of a student, calculate total and print grade if the student has passed in all subjects.

MARKS SHEET		
Name of a student	XYZ	
Class	X	
Subjects	Max Marks	Marks obt
English	100	95
Hindi	100	90
Kannada	100	85
Mathematics	100	90
Social Studies	100	99
Physics	100	90
Chemistry	100	85
Biology	100	95
Total	800	729
Percentage		91.125
Grade		S

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment No 7

Aim: To study Microsoft PowerPoint

➤ **WHAT IS POWER POINT?**

Power point is a complete presentation graphics package. It has the powerful features like power point wizards, toolbars and power point views to create good slides. It has all the tools required to produce a professional looking presentation, such as text handling, outlining, and drawing graphics, clipart and so on. Speaker supports and aids help you to create truly effective presentations. It has wizard, auto layouts, and a complete set of easy to use tools assuring you to have everything you need to share your knowledge with others.

➤ **Menus of power point**

Menu bar has several menus which can be invoked by simply clicking on them, by using a mouse or by using a hot key combination from the keyboard. These menus provide access to different commands of power point. These commands are grouped together in menus. Menu bar displays the list of all these menu groups.

➤ **What is presentation?**

Power point is a good way to communicate ideas simply and effectively. For complex topics that are rich with details, such as a scientific paper or an annual report. Each presentation consists of one more pages or slides, which can contain text, bulleted lists, graphics, charts and other data types.

➤ **Insert a new slide**

To insert a new slide, you can perform any of the following tasks.

1. Insert a slide, go to insert menu and choose slide. 2. Choose a new slide button from standard tool bar.
3. From the power point startup screen, choose blank presentation.
4. If power point is already open, pull down the file menu. Choose new, select blank presentation from the general tab, and click ok.
5. Click the new button on the standard toolbar.

➤ **Delete a new slide**

To delete a slide, make that slides current slide and choose duplicate slide from the edit menu. Slide will be deleted immediately.

➤ **Duplicate a slide**

To duplicate a slide make that slide current slide and choose duplicate slide from the edit menu.

➤ **Creating master slide**

If you want to have certain common items on all the pages without adding them individually to the slides one by one, create a master slide. The items contained in master slide will automatically become the items for all the slides.

➤ **What are presentation graphics?**

Presentation graphics is an application software available for designing charts. You can perform any of the following tasks.

1. Design characters.
2. Arrange the matter in readable form.
3. Add pictures in the charts.
4. Change the appearance of the alphabets on the charts.
5. Print these charts.

➤ **To display slide setup**

In a new presentation, the slides by default have a width of 10 inches, height of 7.5 inches and landscape orientation. These settings can be changed using the page setup commands. The procedure for changing the slide setup is follows:

1. Click on the main menu option.
2. Click on the page setup command, the page setup dialogue box with the default settings appear on the screen.
3. Click on the slides sized for dropdown arrow. 4.
Click on letter paper (8.5*11 in).
5. Click on the portrait radio button.
6. Click on the ok button to change slide settings for every slide in your presentation. The slides will now be 10 inches in height, have a width of 7.5 inches and the orientation will be portrait.

➤ **Saving a presentation**

To save a presentation on disk, click the save button on the standard or choose save option or save as option from the file menu. Option save is to save the file with current name and save as the command to save file with some other name.

➤ **To display a slide show**

A presentation can be displayed on the screen by running a slide show. The slides can be advanced manually or automatically. The procedure for running the slide show is:

1. Click on the slide button. At the bottom of the slide to begin the slide show.
2. Select slide show from the view menu to display a dialog box.
3. One slide is displayed at a time each slide fills the entire screen.
4. Click on the left mouse button or press enter or press page down to move one slide forward.
5. When we reach the last slide in the presentation, power point brings us back to the slide view, or any other view that we are in.
6. Click on file menu option
7. Click on close command to close the presentation.
8. Click on exit command to exit from the power point.

➤ **Adding a clip art to a slide**

1. Choose insert<picture >clipart or double- click a clip artplaceholder to open the insert clip art dialog box.
2. Select the picture you want to insert and click insert menu

Assignment

SET A

1. Use Microsoft PowerPoint to create a slideshow entitled “Me!” Your presentation must contain at least five slides, should be eye-catching and have a creative use of visual and audio effects. Your slides should also be edited for correct use of spelling and grammar. You should discuss:
 - **Your early life** (where you were born, who’s in your family, where you grew up, which elementary school(s) you attended, etc.)
 - **The person that you are now** (hobbies, your favorite music, favorite classes, sports you’re interested in, what makes you *different* than other people, etc.)
 - **What you’d like your future to be** (which high school and college you would like to graduate from, your ideal career after graduation, will you be married?, have children?, etc.)
2. Prepare a power point presentation on Indian Festivals.

SET B

1. Use Microsoft PowerPoint to create a slideshow for Input and Output Devices. Your presentation must contain at least five slides, should be eye-catching and have a creative use of visual and audio effects.
2. Use Microsoft PowerPoint® to create a slideshow entitled “My College!” Your presentation must contain at least five slides, should be eye-catching and have a creative use of visual and audio effects. Give all information of your college.

SET C

1. Make a presentation on “Wild Life” and apply the following:
 - a. Add audio and video effects
 - b. Apply various Color Schemes
 - c. Apply various animation schemes.
 - d. Apply Slide Show

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment No 8

SET A

1. Create a PowerPoint slide show on “Air Pollution”
2. Create a PowerPoint slide show on BCA (Science) Course information.

SET B

1. Create a PowerPoint slide show on “Swachh Bharat Mission” with the contents given below.
 - Select a suitable design template and appropriate slide layouts.
 - Graphics that can enhance your presentation may also be inserted. You can replace standard bullet symbols with other graphics.
 - Add animation effects to the bullet items.
 - Add transition and appropriate sound effects.
2. Use Microsoft PowerPoint to create a slideshow entitled “My Resume!”. Your presentation must contain at least five slides, should be eye-catching and have a creative use of visual and audio effects.

SET C

Use Microsoft PowerPoint to create a slideshow entitled “ M.S.Office!”. Your presentation must contain at least five slides, should be eye-catching and have a creative use of visual and audio effects

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Section III: Shell Commands

Assignment No 9 : Shell Commands

AIM : To study and excute the commands in unix. COMMANDS :

1.Date Command :

This command is used to display the current data and time.

Syntax : \$date \$date +%ch

Options : -

a = Abbrevated weekday. A = Full weekday.

b = Abbrevated month. B = Full month.

c = Current day and time.

C = Display the century as a decimal number. d = Day of the month.

D = Day in „mm/dd/yy“ format h = Abbrevated month day. = Display the hour. L = Day of the year.

m = Month of the year. M = Minute.

P = Display AM or PM S = Seconds

T = HH:MM:SS format u = Week of the year.

y = Display the year in 2

To change the format :

Syntax : \$date „+%H-%M-%S“

2.Calender Command :

This command is used to display the calendar of the year or the particular month of calendar year.

Syntax :

a.\$cal b.\$cal

Here the first syntax gives the entire calendar for given year & the second Syntax gives the calendar of reserved month of that year.

3.Echo Command :

This command is used to print the arguments on the screen .

Syntax :

\$echo

Multi line echo command :

To have the output in the same line , the following commands can be used.

Syntax :

`$echo text`

To have the output in different line, the following command can be used.

Syntax :

`$echo "text >line2 >line3"`

4.Banner Command :

It is used to display the arguments in „#" symbol .

Syntax :

`$banner`

5.'who' Command :

It is used to display who are the users connected to our computer currently.

Syntax :

`$who - option"s`

Options : -

H-Display the output with headers.

b-Display the last booting date or time or when the system was lastely rebooted.

6.'who am i' Command : Display the details of the current working directory.

Syntax :

`$who am i`

7.'tty' Command :

It will display the terminal name.

Syntax :

`$tty`

8.'Binary' Calculator Command : It will change the „\$" mode and in the new mode, arithmetic operations such as +,- ,*,/,%,n,sqrt(),length(),=, etc can be performed . This command is used to go to the binary calculus mode.

Syntax :

`$bc operations ^d $ 1 base -inputbase 0 base - outputbase` are used for base conversions.

Base : Decimal = 10 Binary = 2 Octal = 8 Hexa = 16

9.'CLEAR' Command :

It is used to clear the screen.

Syntax :

`$clear`

10.'MAN' Command :

It help us to know about the particular command and its options & working. It is like „help“ command in windows .

Syntax :

\$man <command name>

11.MANIPULATION Command :

It is used to manipulate the screen.

Syntax :

\$tput <argument> Arguments :

- 1.Clear – to clear the screen.
- 2.Longname – Display the complete name of the terminal.
- 3.SMSO – background become white and foreground become black color. 4.rmso – background become black and foreground becomes white color. 5.Cop R C – Move to the cursor position to the specified location.
- 6.Cols – Display the number of columns in our terminals.

12.LIST Command :

It is used to list all the contents in the current working directory.

Syntax :

\$ ls – options<arguments>

If the command does not contain any argument means it is working in the Current directory.

Options :

- a- used to list all the files including the hidden files. c- list all the files columnwise.
- d- list all the directories.
- m- list the files separated by commas.
- p- list files include „/“ to all the directories.
- r- list the files in reverse alphabetical order.
- f- list the files based on the list modification date. x-list in column wise sorted order.

DIRECTORY RELATED COMMANDS :

1.Present Working Directory Command :

To print the complete path of the current working directory.

Syntax :

\$pwd

2.MKDIR Command :

To create or make a new directory in a current directory .

Syntax :

\$mkdir<directory name>

3.CD Command :

To change or move the directory to the mentioned directory .

Syntax :

\$cd <directory name>

FILE RELATED COMMANDS :

1.CREATE A FILE :

To create a new file in the current directory we use CAT command.

Syntax :

```
$cat ><filename
```

The > symbol is redirectory we use cat command.

2.DISPLAY A FILE :

To display the content of file mentioned we use CAT command without „>“ operator.

Syntax :

```
$cat <filename.
```

Options -s = to neglect the warning /error message.

3.COPYING CONTENTS :

To copy the content of one file with another. If file doesnot exist, a new file is created and if the file exists with some data then it is overwritten.

Syntax :

```
$ cat <filename source> >> <destination filename>
```

```
$ cat <source filename> >> <destination filename> it is avoid overwriting.
```

Options : -

-n content of file with numbers included with blank lines.

Syntax :

```
$cat -n <filename>
```

4.SORTING A FILE :

To sort the contents in alphabetical order in reverse order.

Syntax :

```
$sort <filename >
```

Option : \$ sort -r <filename>

5.COPYING CONTENTS FROM ONE FILE TO ANOTHER :

To copy the contents from source to destination file . so that both contents are same.

Syntax :

```
$cp <source filename> <destination filename>
```

```
$cp <source filename path > <destination filename path>
```

6.MOVE Command :

To completely move the contents from source file to destination file and to remove the source file.

Syntax :

```
$ mv <source filename> <destination filename>
```

7.REMOVE Command :

To permanently remove the file we use this command .

Syntax :

```
$rm <filename>
```

8.WORD Command :

To list the content count of no of lines , words, characters .

Syntax :

\$wc<filename>

Options :

-c – to display no of characters.

-l – to display only the lines.

-w – to display the no of words.

9.LINE PRINTER :

To print the line through the printer, we use lp command.

Syntax :

\$lp <filename>

10.PAGE Command :

This command is used to display the contents of the file page wise & next page can be viewed by pressing the enter key.

Syntax :

\$pg <filename>

Section IV: HTML

Assignment No 10: Introduction to HTML

Internet and Web:

The Internet is generally defined as a global network connecting millions of computers.

The Internet is **not** synonymous with World Wide Web. The Internet is a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Browsers running on client machines request documents provided by servers. Browsers are so called because they allow the user to browse through the documents available on the web servers. A browser initiates the communication with a server, requesting for a document. The server that is continuously waiting for a request, locates the requested document and sends it to the browser, which displays it to the user. The most common protocol on the web is Hyper Text Transfer protocol (HTTP). The most commonly used browsers are Microsoft Internet Explorer (IE), Netscape browser and Mozilla. The most commonly used web servers are Apache and Microsoft Internet Information server (IIS).

- **HTML :**

HTML stands for Hyper Text Markup Language

Hyper Text Markup Language is a simple markup language used to create platform-independent hypertext documents on the World Wide Web.

Most hypertext documents on the web are written in HTML. HTML is the standard markup language for creating Web pages.

HTML describes the structure of Web pages using markup.

HTML elements are the building blocks of HTML pages.

HTML elements are represented by tags.

HTML tags label pieces of content such as "heading", "paragraph", "table", and so on.

Browsers do not display the HTML tags, but use them to render the content of the page

- **Steps to write HTML program : -**

1. Open a notepad or text editor application and write the html code.
2. The <html> <body> Program Body </body></html>
3. The file will be saved with extension .html or .htm.
4. Execute the program by using compatible web browser.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title> My First HTML Page</title>
</head>
<body>
    Welcome to HTML
</body>
</html>
```

Explanation of example:

- The <!DOCTYPE html> declaration defines this document to be HTML5.
- The <html> element is the root element of an HTML page.
- The <head> element contains meta information about the document.
- The <title> element specifies a title for the document.
- The <body> element contains the visible page content.

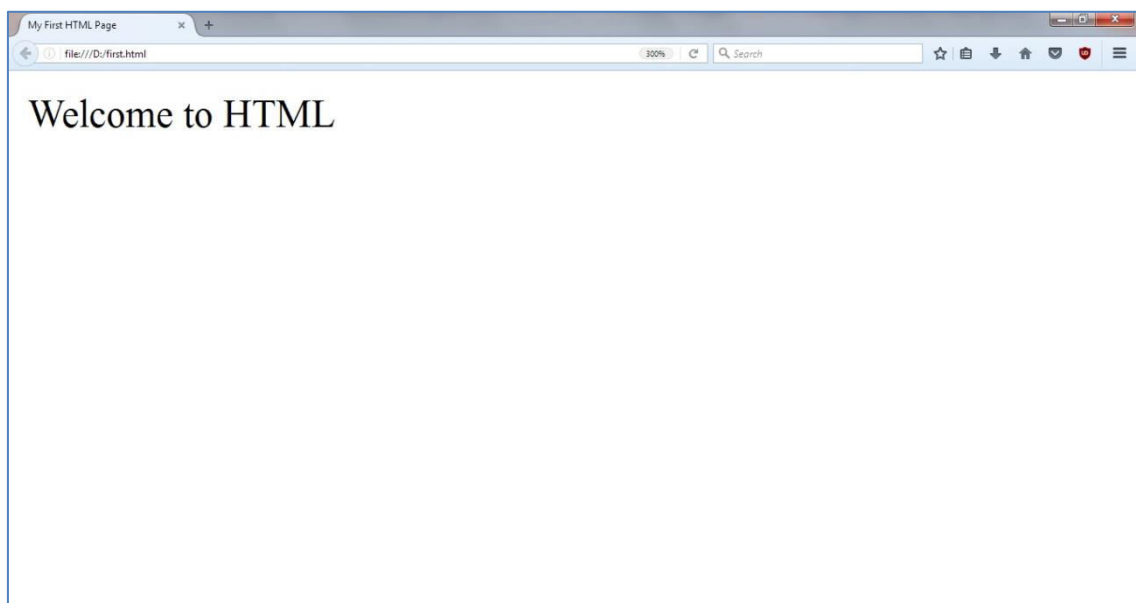
HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

- HTML tags normally come **in pairs** like <p> and </p>
- The first tag in a pair is the **start tag**, the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

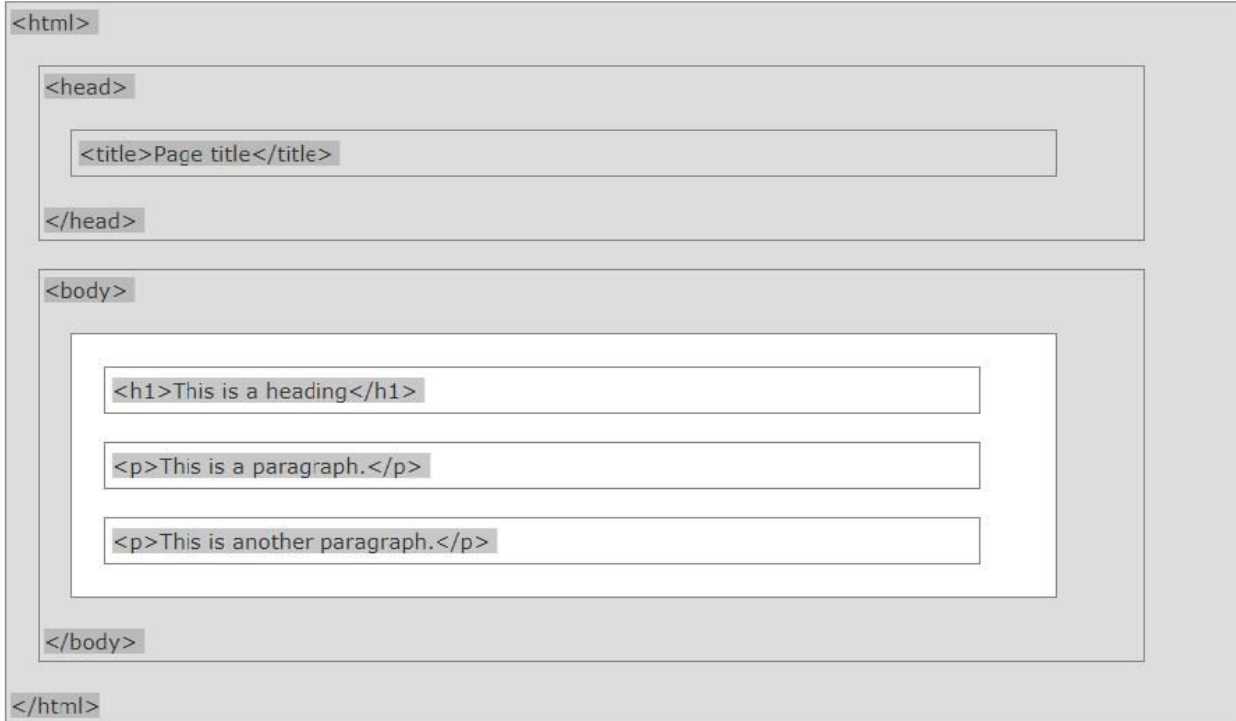
Web Browsers

- The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.
- The browser does not display the HTML tags, but uses them to determine how to display the document:



HTML Page Structure

Below is a visualization of an HTML page structure:

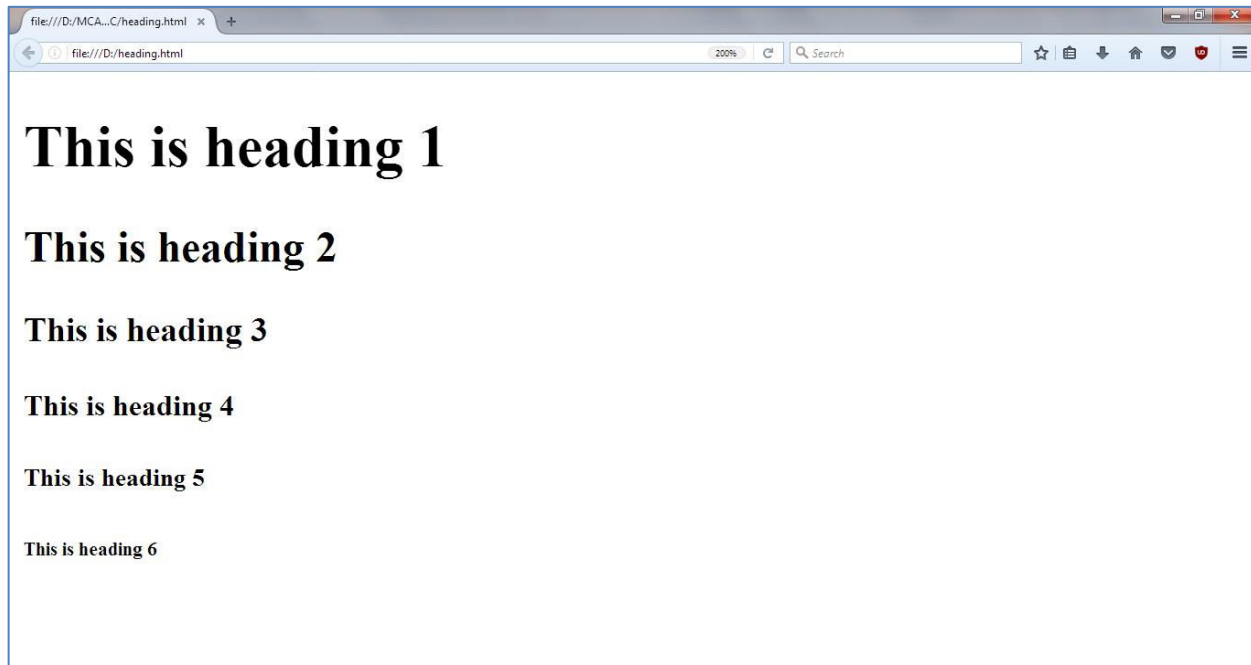


HTML Headings

HTML headings are defined with the **<h1>** to **<h6>** tags.

<h1> defines the most important heading. **<h6>** defines the least important heading:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>This is heading 1</h1>
    <h2>This is heading 2</h2>
    <h3>This is heading 3</h3>
    <h4>This is heading 4</h4>
    <h5>This is heading 5</h5>
    <h6>This is heading 6</h6>
  </body>
</html>
```



- Some HTML tags required to design simple web pages are given below

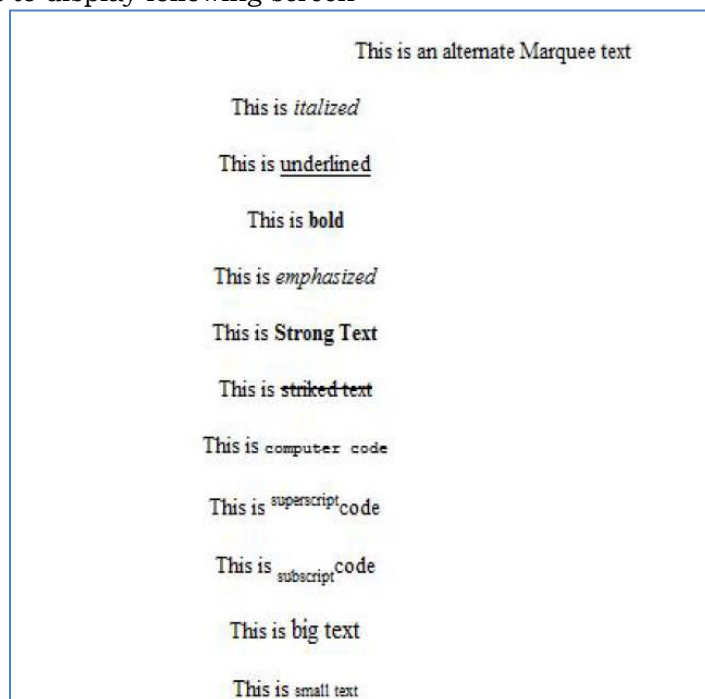
Tag	Description	Attributes	Example
<!-- ... -->	Allows one to insert a line of browser invisible comments in the document		<!-- Starting my first web page --!>
<body> </body>	Every web page needs a body in which one can enter web page content	background= designates a file to be displayed as background bgcolor= "#(hexadecimal color code)" sets the background color text= "#(hexadecimal color code)" sets the color of plain text. Text color default is black.	<body bgcolor="#00FF00" text="#FF0000">Page with Green Color and red Text</body> Format of color number is RRGGBB, so if we write 00FF00 we mean (red=0, green=255, blue=0)
 	A single tag used to break lines	clear=all left right Breaks the text and resumes the next line after the specified margin is clear.	line is broken
<p>	Used as paragraph. A single tag used to break text as paragraph. Breaking text with the <p> tag adds vertical spacing		<p> break the line</p> <p>adding extra space </p>
	Used to represent text in bold		This text will appear bold
<u></u>	To make text appear underlined		This is <u>underlined tag </u>
<i></i>	To make text appear italic		This is <i>italized </i>
<center></center>	Centers enclosed text		<center> Text is centered </center>
<big></big>	Sets the type one font size larger than the surrounding text		<big> This will appear one size big</big>
<small></small>	Sets the type one font size smaller than the surrounding text		<small> This will appear one size small</small>
	Formats enclosed text as superscript.		X^{<small> 2}</small>
	Formats enclosed text as subscript		H_{<small> 2}</small>O
<marquee></marquee>	Creates a scrolling text marquee area.	1.align=top middle bottom - Aligns the marquee with the top, middle or bottom of the neighboring text line.	<marquee align=top behaviour =slide bgcolor="pink" direction=right height=20 hspace =5 > scrolling

		2.behavior=scroll slide alternate - Specifies how the text should behave. -Scroll is the default setting and means the text should start completely off one side, scroll all the way across and completely off, then start over again. -Slide stops the scroll when the text touches the other margin. - Alternate means bounce back and forth within the marquee. 3.bgcolor="#rrggbb" or color name	all the way from one end to other </marquee>
	loads an inline image	src= " text" Provides the URL of the graphic file to be displayed alt="text" Provides alternate text if the image cannot be displayed. height= <i>number</i> Specifies the height of the image in pixels. width= <i>number</i> Specifies the width of the image in pixels.	

Assignment

Set A:

1. Write a HTML script to display following screen



2. Create an html page with all the different text styles (bold, italic and underlined) and its combinations on separate lines. State style of each line in its text.
3. Create an html page containing the polynomial expression as follows:
$$A_0 + A_1X + A_2X^2 + A_3X^3$$

Set B:

1. Write a HTML script which will demonstrate all attributes of the text tag.
2. Write a HTML script that will use image as a background and move one image from top to bottom another image from right to left, on screen.

Set C:

1. Create an html page with following specifications
 - a. Title should be about myCity
 - b. Place your City name at the top of the page in large text and in blue color
 - c. Add names of landmarks in your city each in a different color, style and typeface
 - d One of the landmark, your college name should be blinking
 - e. Add scrolling text with a message of your choice.
 - f . Add some image at the bottom
2. Create an html page with red background with a message “warning” in large size blinking. Add scrolling text “read the message” below it.

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment No. 11: Working with List and Hyperlink

In addition of basic tags HTML also supports some of the features that we will discuss in this assignment.

1. List: Lists are a great way to provide information in a structured and easy to read format.

There are two types of lists provided in HTML:

- 1. Numbered List (Ordered List):** An ordered list is used when sequence of list items is important.
- 2. Bulleted List (Unordered List):** An unordered list is a collection of related items that have no special order or sequence.

Tags used to create lists are given in the following table:

Sr. No	Tag	Description	Attributes	Example
1.	 	Specify the list item.		
2.		The tag is used to specify the ordered list. It starts at 1. And always incremented by 1.	Type = a/A/i/I/1 Sets the numbering style to a,A,i, I,1 default 1 start = "A" Specifies the number or letter with which the list should start.	 Cats Dogs <ol start="5"> Cats Dogs <ol reversed> BCA MCA B.Sc
3.	 	The tag is used to specify the unordered list.	Type = disc/square/circle Specifies the bullet type.	<ul type=circle> ul - unordered list ol - ordered list menu - menu list
4.	<dl> </dl>	The <dl> tag is used to specify the definition list.		<dl> <dt> Bangalore </dt> <dd> -Capital City of Karnataka </dd> <dt> Mumbai</dt> <dd> -Capital city of Maharashtra </dd> </dl>
5.	<dt> </dt>			
6.	<dd> </dd>			

2. Hyperlinks : Hyperlink is a specialized feature of HTML. Instead of clicking through sequentially organized pages, a hypertext user clicks specially highlighted text called 'hyperlink'. Hyperlinks are technically known as anchors. They are usually visible in blue underlines.

- **Tags used to add Hyperlinks in html document are given in the following table.**

Sr. No	Tag	Description	Attributes	Example
1	<A>	Adds an anchor or hyperlink	href= "url" specifies the url of the target page.	<html><body> Click here to search</body></html>

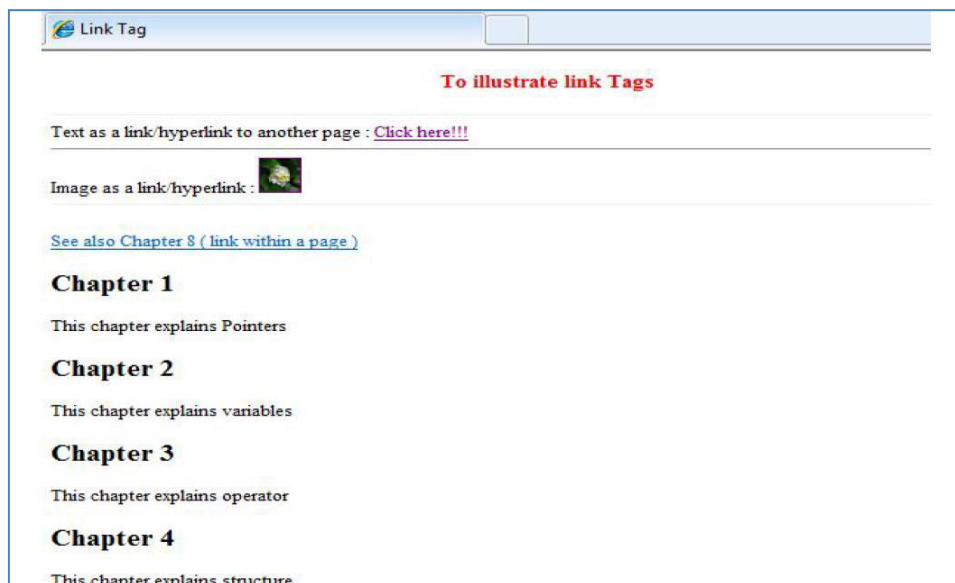
Assignment

Set A

- Write the HTML code which generates the following output.
 - Coffee
 - Tea
 - Black Tea
 - Green Tea
 - Africa
 - China
 - Milk
- Create an html5 page with which generates the following output.
 - Cold Drinks
 - Juices
 - Lemon Water
 - Soda Water
 - Hot Drinks
 - Tea
 - Coffee
 - Milk

Set B

- Create an HTML5 program for unordered list of flowers with its color in nested list. Modify it to change the shape of the bullet, also reduce the size of bulleted items one smaller than the heading.
- Write a HTML script to display following screen



- Write the HTML code for generating the form as shown below

City Gallery

- London
- Paris
- Tokyo

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants. Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.

Set C

1. Create an html page with appropriate Heading and other information. Add an ordered list of your educational qualifications. For each course make a nested list that contains, university or board name, the year and the percentage scored. Add link to university site where university name appears. Add your college photograph and message in a separate web page
2. Create a Web Site with following specifications
 - a) Title should be about MY CITY
 - b) Put image of your city map in the background
 - c) Place popular college name of your city at the bottom in smaller size
 - d) Add names of historical places in a different color, style and typeface
 - e) Add scrolling text with a message of your choice
 - f) Add photo of historical place at the top.

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

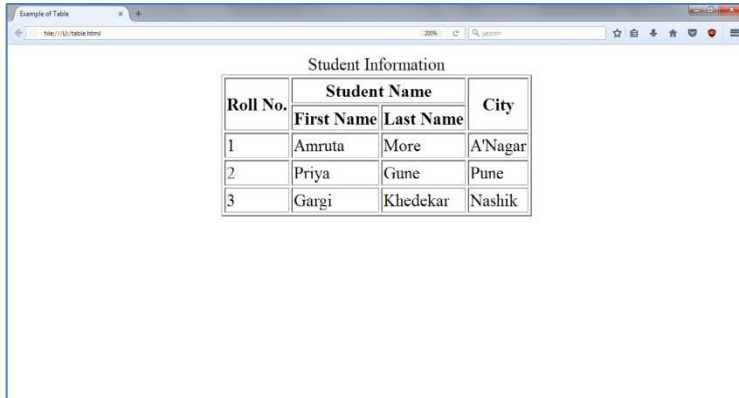
Assignment No. 12: Working with Table

Table: A table is a two dimensional matrix, consisting of rows and columns. HTML tables are intended for displaying data in columns on a web page. Table contains information such as text, images, forms, hyperlinks etc.

- **Tags used to create Tables are given in the following:**

Sr. No	Tag	Description	Attributes
1.	<table> </table>	Used to create a table	1. Border= number Used to draw a border around to the table row and cells of width equal to the specified number. By default no border i.e. number=0. 2.width= number specifies width of the table 3.cellspacing=number Specifies the amount of cell space between the table cells. By default 2. 4.cellpadding=number Sets the amount of cell space in number of pixels between the cell border and its contents. Default is 2 5.bgcolor="rrggbb" sets background color of table 6.bordercolor="rrggbb" sets border color of table. 7. align=left right center Aligns the table. Default alignment is left. 8.frame=void above below hside lhs rhs vside box border Tells the browser where to draw borders around the table
2.	<tr></tr>	Creates a table row.	
3.	<th></th>	Creates table heading	
4.	<td></td>	Data cells are inserted in a row of the table	1.rowspan=number Specifies number of rows a cell should span. 2.colspan=number Specifies number of columns a cell should span 3.cellspacing=number Specifies space between two cells 4.cellpadding=number Specifies space between text and columns

Example



Roll No.	Student Name		City
	First Name	Last Name	
1	Amruta	More	A'Nagar
2	Priya	Gune	Pune
3	Gargi	Khedekar	Nashik

```
<html>
<head>
<title> Example of Table</title>
</head>
<body>
<center>
<table border="1">
<caption>Student Information</caption>
<tr>
<th rowspan=2>Roll No. </th>
<th colspan=2>Student Name </th>
<th rowspan=2>City </th>
</tr>
<tr>
<th>First Name</th>
<th>Last Name</th>
</tr>
<tr>
<td>1 </td>
<td>Amruta </td>
<td>More </td>
<td>A'Nagar </td>
</tr>
</table>
</center>
</body>
</html>
```

Assignment

Set A

1. Create an html5 page with which generates the following output

Player Name	No. of Matches played	No. of innings played	Score	Average
Sachin Tendulkar	325	320	14545	45.43
Rahul Dravid	310	280	10450	38.12
Anil Kumble	312	160	400	10
M.S. Dhoni	250	200	4000	30

2. Create an html5 page with which generates the following output.

List of Books			
Book No	Book Name	Price	
		Rs.	Paise
1	Java programming	350	50
2	C programming	250	00

Set B

1. Create an html page to generate your class time table
2. Create an html page to generate Bus Schedule from Aurangabad to Pune and from Pune to Kolhapur.

Set C

1. Create an html5 page with which generates the following output

Country	Population (In Crores)	
INDIA	1998	85
	1999	90
	2000	100
USA	1998	30
	1999	35
	2000	40

2. Create an html5 page with which generates the following output

Train name	Starting place	Destination place	Time		Fare
			Arrival	Departure	

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment No.13 Working with Frames

Frame: Using frames, one can divide the screen into multiple scrolling sections, each of which can display a different web page into it. It allows multiple HTML documents to be seen concurrently

Inline Frame: It is a new frame tag introduced in HTML5. It is having same properties and attribute options as in <FRAME> tag. An <iframe> tag is used to display a web page within a web page. Inline frames can be included within the text block in HTML5 document.

- Tags used to add frame and iframe in html document are given in the following table.

Sr. No	Tags	Description	Attributes	Example
1.	<frame> </frame>	Splits the webpage into frames.	1.rows=number Divides the browser screen into horizontal section. 2.cols=number Divides the browser screen into vertical section. The number written in the rows and cols attribute can be given as absolute number value or an asterisk (*) can be used to indicate the remaining space.	<frameset rows = "10%,40%,*">
2.	<frameset> </frameset>	Defines a single frame in a frameset.	1. name=text assigns name to a frame 2. noresize- prevents to resize the frame 3. src=url Specifies initial html file to be displayed as home screen 4.bordercolor="rrggbb" Specifies the border color of the frame	<html> <frameset rows="50%,*"> <frameset cols="50%,*"> <frame src="welcome.html" name="frm1"> </frameset> <frame src="menu.html" name="mnu"> </frameset> </html>
3.	<iframe> </iframe>	This is used to specify inline frame. It allows us to embed another document in current html document	1.src=url of initial iframe 2.name=name of the iframe 3.longdesc=long description of frame 4.width=number/pixel 5.height=number/pixel 6. align=[top middle bottom left right center] (frame alignment, pick two, use comma)	<!DOCTYPE html> <html> <body> <iframe src="https://www.dypacs.com"> <p>Your browser does not support iframes.</p> </iframe> </body> </html>

Example:

First page.html

Thirdpage.html

Secondpage.html

```
<html>
<head>
<title> First Page</title>
</head>
<body>
<h1>This is First Web Page</h1>
</body>
</html>
```

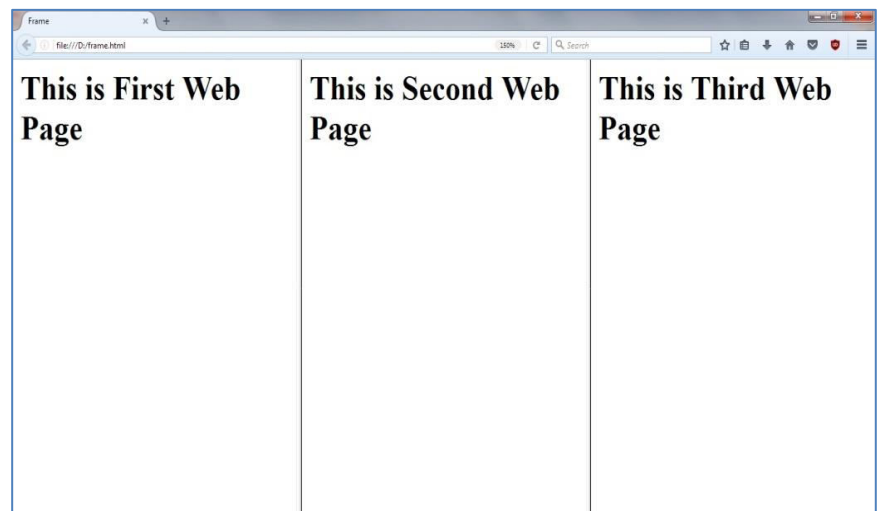
```
<html>
<head>
<title> Second Page</title>
</head>
<body>
<h1>This is Second Web Page</h1>
</body>
</html>
```

```
<html>
<head>
<title> Third Page</title>
</head>
<body>
<h1>This is Third Web Page</h1>
</body>
</html>
```

Frame.html

Output File

```
<html>
<head>
<title> Frame </title>
</head>
<frameset cols="*, *, *>
<frame src="f1.html">
<frame src="f2.html">
<frame src="f3.html">
</frameset>
</html>
```



Assignment

Set A:

1. Write an html script to generate following output.

Page 1
Page 2
Page 3

2. Write an html script to demonstrate inline frame.
3. Write an html script to generate following output.

This is a header.	
Look in the box at the right for some information.	Here is some information.
This is a footer.	

Set B:

1. Create an html page with which generates the following output. Divide the frame into different sections as shown below and add appropriate html files to each frame.


First Frame : Name and Address	
Second Frame	Third Frame
Bulleted list of qualifications	Links to Favorite sites

2. Write an html script to generate following output.

1		
2	3	4
5		

Set C

1. Write an html script to generate following output.

Welcome to Wipro Company	
	<ul style="list-style-type: none">a. About Usb. Employeesc. Careers

2. Write an html script to generate following output.

<p>This is a first paragraph</p> <p>This is another paragraph</p>	<p>This cell contains a table:</p> <table><tr><td>A</td><td>B</td></tr><tr><td>C</td><td>D</td></tr></table>	A	B	C	D
A	B				
C	D				
<p>This cell contains a list</p> <ul style="list-style-type: none">BatBallStumps	<p>Good Bye!!</p>				

3. Create an html page with appropriate frames containing Heading and other Information. Add an ordered list of your educational qualifications. For each course make a nested list that contains, university or board name, the year and the percentage scored

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor

Assignment : 14 Working with Forms

HTML Forms:

HTML5 provides better & more extensive support for collecting user inputs through forms. A form can be placed anywhere inside the body of an HTML document. You can have more than one form in the document

- Tags used to add input forms are given in the following table.

Sr.No.	Tag	Description	Attributes	Example
1	<code><form></code> <code></form></code>	Creates a form	1. <code>action="URL"</code> Url specifies next form that receives the control 1. <code>method="post/get"</code> Specifies by which method form transfers data to server	<code><!DOCTYPE html></code> <code><html></code> <code><body></code> <code><form method="post" action="next.html"></code> <code>

</code> <code><input type="submit" value="Submit"></code> <code></form></code> <code></body></code> <code></html></code>
2	<code><input></input></code>	This is used to use input control on html form	1. <code>Name=text</code> Used to name the field 2. <code>maxlength=number</code> Used to specify number of input character allowed in field 3. <code>size=number</code> The width of the input control in pixel 4. <code>type="(checkbox/hidden/ radio/reset /submit /text /image)"</code> value to be submitted with the form (for a checkbox or radio button) or label (for Reset or Submit buttons)" 5. <code>src="source file for an image",\</code> 6. <code>checked</code> indicates that checkbox or radio button is checked. 7. <code>align="(texttop/absmiddle /baseline/bottom)"</code>	<code><label for>enter your name</label></code> <code><input type="text" name="nm" width=20></code> <code><input type="radio" name="gender" value="male" checked>Male</code> <code><input type="radio" name="gender" value="Female" >Female</code> <code><input type="checkbox" name="chess" value="chess"> Chess</code> <code><input type="checkbox" name="Poker" value="Poker"> Poker</code>
3	<code><select></code> <code></select></code>	Defines and displays a set of optional list items from	1. <code>name=" (name to be passed to the script as part of name/value pair)"</code> 2. <code>rows="no. of rows"</code>	<code>
</code> Age Between: <code><select name="age" size=1></select></code>

		which the user can select one or more items.	3.cols="(no. of cols.)"	
4.	<textarea> </textarea>	used for multiline text entry	1.name=name of data field 2.size=#of items to display multiple allows multiple selections	<textarea rows=10 columns=40> </textarea>
5	<option>	indicates a possible item within a select widget	1.selected=default selection 2.value="data submitted if this option is selected"	<select name="age" size=1> <option selected>21-30 <option>31-40 </select>

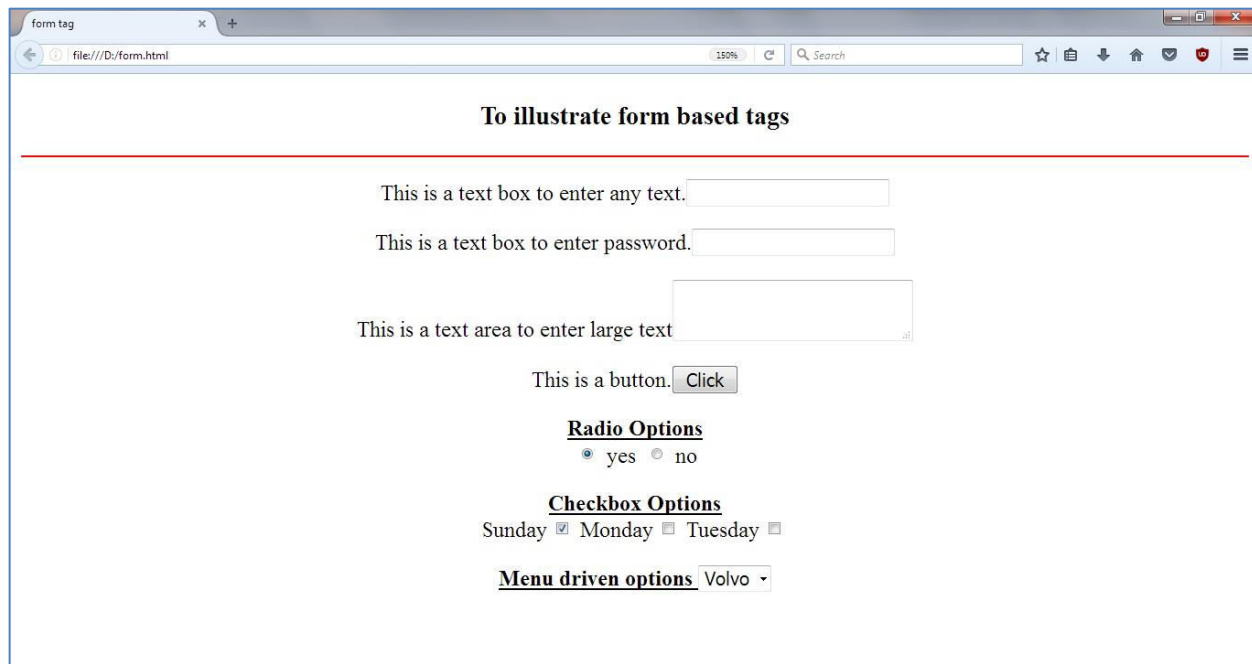
Example:

```

<html>
<head>
<title> form tag </title>
</head>
<body>
<center>
<h3 align="center">To illustrate form based tags</h3> <hr color="red">
<form action="">
<p>This is a text box to enter any text.<input type="text" >
<p>This is a text box to enter password.<input type="password" >
<p>This is a text area to enter large text<textarea> </textarea>
<p>This is a button.<input type="button" Value="Click" >
<p><b><u>Radio Options</u></b><br>
<input type="radio" name="y" checked> yes
<input type="radio" name="n" > no </p>
<p><b><u>Checkbox Options</u></b><br>
Sunday<input type="checkbox" checked >
Monday<input type="checkbox" >
Tuesday<input type="checkbox" >
</p>
<p><b><u>Menu driven options </u></b>
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select> </p>
</form>
</center>
</body>
</html>

```

Output



The screenshot shows a web browser window with a single tab titled 'form tag'. The address bar shows the file path 'file:///D:/form.html'. The page content is as follows:

To illustrate form based tags

This is a text box to enter any text.

This is a text box to enter password.

This is a text area to enter large text

This is a button.

Radio Options
☒ yes ☐ no

Checkbox Options
Sunday ☒ Monday ☐ Tuesday ☐

Menu driven options

Assignment

Set A:

1. Write an HTML code for creating a following form

Enter the details

Enter your nationality:

Enter your age:

2. Write an HTML code for creating Login Form. Login form contains Username, Password, Confirm Password fields with Submit Button.

Set B:

1. Write an HTML code for creating a following form

Choose your favourite ice cream flavour

How would you like to have it?

☐ CUP ☐ CONE ☒ BAR

How Many people would you like to serve?

Tell Us something about your self

To clear the contents click.

Flavour list: Vanilla, Pistachio, Chocalate, Mango, Santra Mantra

2. Write an HTML script for creating Railway Reservation form.

Set C:

1. Write an HTML code for creating a form, which accepts the birth date from the user in a textbox and displays the day of the week in a message box on the click of a button.
2. Design an html form to take the information of a article to be uploaded such as file path, author name, type (technical, literary, general), subject topic (to be selected from a list) etc. One should provide button to Submit as well as Reset the form contents.

Assignment Evaluation

0: Not Done []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

Signature of Instructor