PPS Unit wise Question Bank

Short Answer Questions:

Q.No	**Questions**
	UNIT 1
1	Write about Components of a computer system?
2	Differentiate between Compiler and Interpreter?
3	Differentiate between Primary and Secondary memory?
4	What is an Algorithm?What are it's advantages?
5	Write an algorithm to find greatest amongst three given numbers?
6	What is FlowChart?List the notations used along with it's advantages?
7	Draw a FlowChart to find the greatest amongst three given numbers?
8	What is C programming language?List the advantages of it?
9	Define variable and mention the rules to be followed while creating a variable?
10	Write a c program to find the sizeof each data type?
11	What is Operator?Mention the different kinds of operators in C?
12	What are Storage Classes?Mention their advantages?
13	Write a c program to implement type conversion?
14	Write a c program to accept command line arguments?
15	What are control statements?Write a c program to implement do while loop?
16	Differentiate between while loop and do-while loop?
17	What are command line arguments?
	UNIT 2
1	What is an Array?How do we declare an array?Mention the different types of array along with example?
2	Write about two dimensional array with example?
3	Define string?What are the different ways to accept a string?
4	Write a c program to handle string as array of characters?
5	Write the syntax of the following i)strlen ii)strcat iii)strcpy iv)strstr
6	Define structure and mention the syntax of it along with an example?
7	Define union and mention the syntax of it along with an example?
8	Differentiate between structures and unions?
9	Define pointers and mention their advantages?
10	Write a c program to implement enumerated data type?
11	List the usage of self referential structures in Linked List?
12	Write a c program to compare two strings?
	UNIT 3
1	What is preprocessor directive?Mention its usage?
2	Write about the following

	i)#include ii)#define iii)#undef
3	Write a c program to implement #ifdef directive?
4	Write a c program to implement #ifndef directive?
5	Differentiate between text and binary files?
6	What are the different modes of operations on text file?
7	Write a c program to append data to existing file?
8	Write a c program to implement fseek()?
9	Write a c program to implement ftell()?
10	Write a c program to implement rewind()?
11	Write the syntax of functions used in text and binary file operations?
UNIT 4	
1	What is function?How to declare a function along with an example?
2	Differentiate between formal arguments and actual arguments?
3	Differentiate between call by value and call by reference?
4	Writea c program to find factorial of number using recursion?
5	Write a c program to find Fibonacci series using recursion?
6	Write a c program to find if a number is prime number or not?
7	Write a c program to find Armstrong number?
8	What is recursion?What are its advantages?
9	What is dynamic memory allocation and mention the need for it?
10	Write the syntax of the following i)malloc ii)calloc iii)realloc iv)free
UNIT 5	
1	What is the need for the algorithm?
2	Write an algorithm to find roots of quadratic equation?
3	Write an algorithm to find prime number?
4	What is searching technique?Mention its type along with advantages?
5	Write about linear search technique and steps followed for it?
6	Write about Binary search technique and steps followed for it?
7	Define sorting technique?Write an algorithm to perform bubblesort?
8	Explain selection sort algorithm ?
9	Explain insertion sort algorithm?
10	Define Time Complexity?Define Space Complexity?
11	What is quick sort algorithm?
12	What is merge sort algorithm?

Long Answer Questions

Q. No	Questions
UNIT 1	
1	What is flow chart? Explain the different notations used in flow chart?
2	What is an Algorithm? Explain the characteristics of algorithm?
3	Write a c program to perform arithmetic operations using switch case?
4	Explain the general structure of c program along with an example?
5	Write a c program to implement ternary operator?
6	What is an operator? Explain different types of operator along with an example?
7	Explain operator precedence in detail?
8	Write a c program to implement bitwise operator ?
9	Explain about storage classes in detail along with an example program?
10	Explain conditional statements in c along with an example?
11	Write a c program to implement logical operators ?
12	Write a c program to find Armstrong number?
13	what are command line arguments? explain with an example program?
14	Write a c program to implement implicit and explicit type conversion?
15	Write a c program to implement relational operators?
16	Explain the process of accepting input and output through scanf and printf?
17	What is goto statement? Why it is not preferred in programs along with an example?
18	Write the syntax of the following: i)if ii)if else iii)if..else..if iv)while v)dowhile vi)for

19	Write a c program to print the following :
	`1`
	`2 2`
	`3 3 3`
	`4 4 4 4`

UNIT 2	
1	Write a c program to calculate sum of all elements in an array?
2	What is an Array?List the different types of array?Write a c program to display the elements of the array?
3	Explain in detail creation and manipulation of elements in an array?
4	Write a c program to calculate addition oftwo matrices?
5	Write a c program to multiply two matrices?
6	Define Strings?Explain initialization of String and accepting the string along with an example program?
7	Write a c program to pass strings to functions?
8	Write about the following: i)strlen ii)strcat iii)strcpy iv)strstr
9	Write a c program to implement strcpy and strlen functions?
10	Write a c program to implement strlwr function?
11	What are structures?Write a c program to initialize and display the elements of

	structures?
12	What are unions? Write a c program to initialize and display the elements of structures?
13	Explain Array of Structures in detail along with an example program?
14	Write a c program to implement enumeration data type?
15	Define pointer?Write a c program to perform addition of two numbers using pointers?
16	Write a c program to implement self referential structures?
UNIT 3	
1	With a neat diagram explain the program process flow with each steps?
2	Write a c program to find area of circle using #define preprocessor?
3	Explain #define preprocessor directive in detail?
4	Write a c program to implement #undef directive?
5	Explain the following directives along with an example program i)#ifdef ii)#ifndef iii)#if iv)#else
6	Explain the following directives along with an example program i)#elif ii)#error iii)#pragma
7	What is a File?Explain the different types of Files along with their mode of operations?
8	Write a c program to read and write the data to text file using fprintf and fscanf function?
9	Explain the process of reading and writing information to binary file along with an example program?
10	Explain fseek function in detail along with an example program?
11	Explain ftell and rewind function through an example program?
UNIT 4	
1	Explain functions in detail along with an example program?
2	Explain actual arguments and formal arguments with the help of program?
3	Write a c program to swap two numbers using call by reference?
4	Differentiate between Call By Value and Call By Reference in detail?
5	Write a c program to pass arrays to a function?
6	What is recursion?Explain different types of recursion along with their advantages and disadavantages?
7	Write a c program to find the factorial of a number using recursion?
8	Explain dynamic memory allocation in detail along with an example program?
UNIT 5	
1	Write an algorithm to perform arithmetic operation?
2	Write an algorithm to find the greatest among three numbers?
3	Write an algorithm to perform binary search?
4	Write an algorithm to perform linear search?
5	Write an algorithm to find roots of quadratic equations?
6	Explain linear search along with an example program?

7	Explain binary search along with an example program?
8	What is sorting?What is the need for sorting?Mention the different types of sorting algorithm and write a c program to perform bubble sort?
9	Write a c program to perform merge sort?
10	Write a c program to perform insertion sort?
11	Write a c program to perform selection sort?
12	Write a c program to perform quick sort?
13	Explain the process of analyzing the algorithm?

UNIT-1

Short Answer Questions for 2 Marks:

1. **Write about Components of a computer system?**

There six components of computer system

1.Input Unit

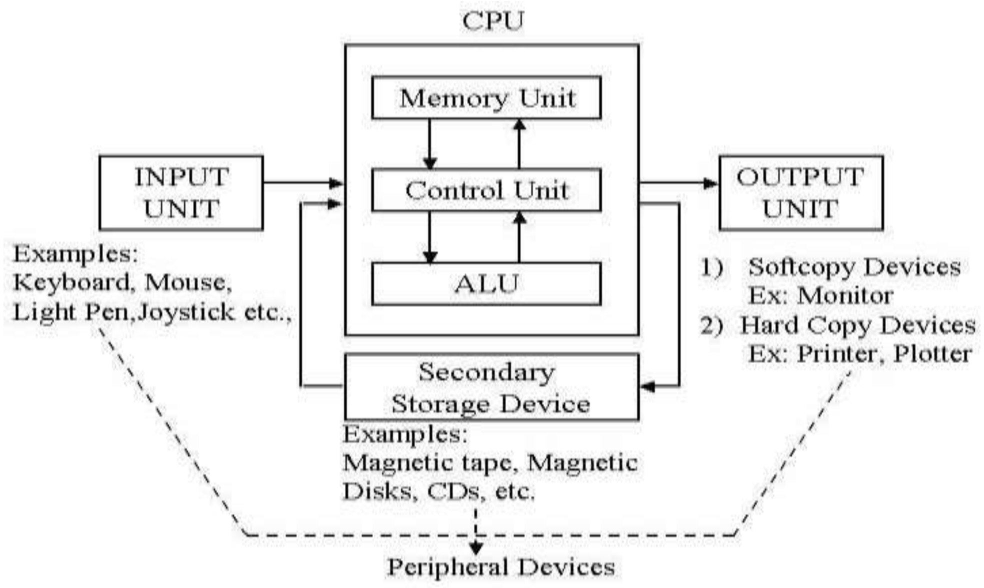
2.Output Unit

3.Storage Unit

4.Central Processing Unit (CPU)

5.Arithmetic and Logic Unit (ALU)

6.Control Unit


CPU
Memory Unit
Control Unit
ALU
INPUT UNIT
Examples: Keyboard, Mouse, Light Pen,Joystick etc.,
OUTPUT UNIT
1) Softcopy Devices Ex: Monitor
2) Hard Copy Devices Ex: Printer, Plotter
Secondary Storage Device
Examples: Magnetic tape, Magnetic Disks, CDs, etc.
Peripheral Devices

2. **Differentiate between Compiler and Interpreter?**

Compiler	**Interpreter**
Scans the entire program and translates it as a whole into machine code.	Translates program one statement at a time.

It takes a large amount of time to analyze the source code but the overall execution time is comparatively faster.	It takes less amount of time to analyze the source code but the overall execution time is slower.
Programming languages like C, C++, Java use compilers.	Programming languages like Python, Ruby use interpreters.

3. Differentiate between Primary and Secondary memory?

Primary Memory	**Secondary Memory**
It is the main memory where the data and information are stored temporarily	It refers to the external memory where data is stored permanently
Data is directly accessed by the processing unit.	Data cannot be accessed directly by the processor
It is a volatile memory meaning data cannot be retained in case of power failure	it is a non volatile memory so data can be retained even after power failure

4. What is an Algorithm?What are it's advantages?

An Algorithm is a step by step procedure for solving a given problem.

Advantages:

It is not dependent on any programming language,so it is easy to understand.

A problem is divided into smaller steps, which makes it easier to convert it into a program.

5. Write an algorithm to find greatest amongst three given numbers?

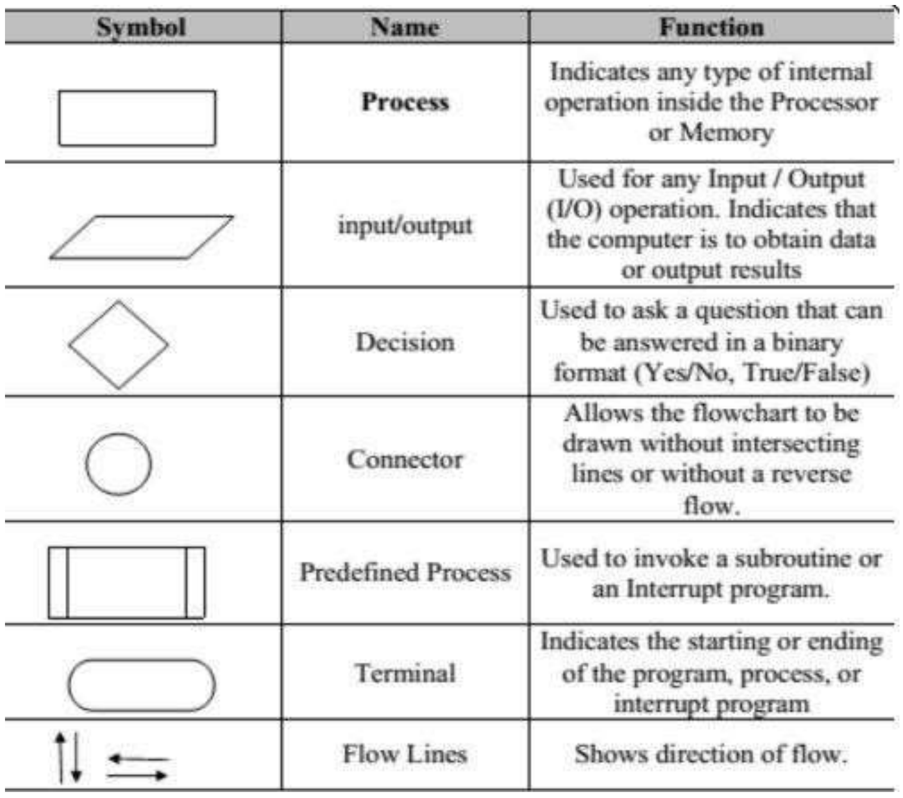
```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a>b
If a>c
```

```
Display a is the largest number.
Else
Display c is the largest number.
Else
If b>c
Display b is the largest number.
Else
Display c is the greatest number.
Step 5: Stop
```

Short Answer Questions for 3 Marks:

6. **What is FlowChart?List the notations used along with it's advantages?**

A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes.



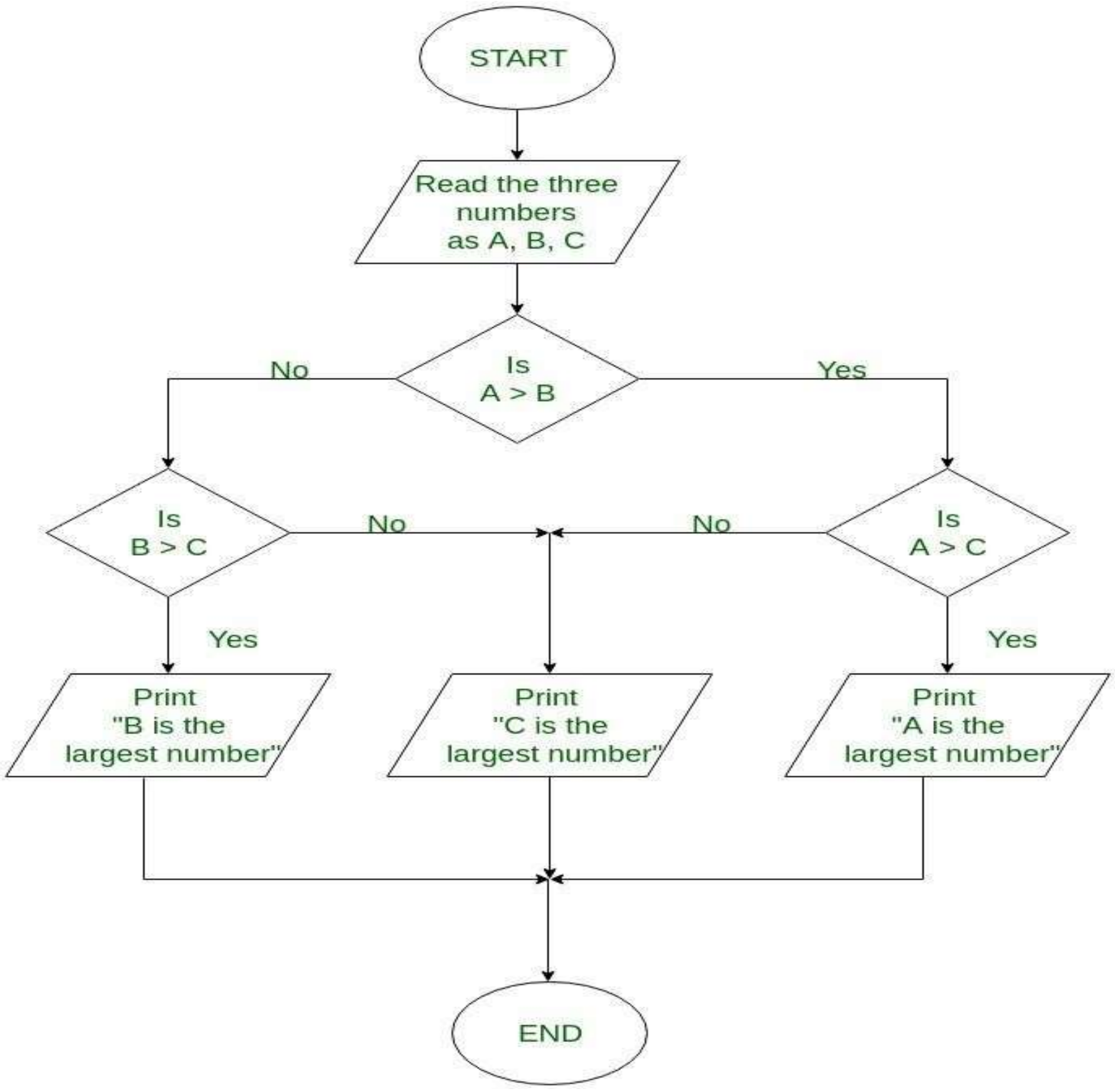
Symbol	Name	Function
▭	**Process**	Indicates any type of internal operation inside the Processor or Memory
▱	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
◇	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
○	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
◫	Predefined Process	Used to invoke a subroutine or an Interrupt program.
⬭	Terminal	Indicates the starting or ending of the program, process, or interrupt program
⇅ ⇄	Flow Lines	Shows direction of flow.

Advantages:

Communication: Flowcharts are better way of communicating the logic of a system to all concerned or involved.

• Effective analysis: With the help of flowchart, problem can be analysed in more effective way therefore reducing cost and wastage of time.

7. Draw a FlowChart to find the greatest amongst three given numbers?



START
Read the three numbers as A, B, C
Is A > B
No
Yes
Is B > C
No
No
Is A > C
Yes
Yes
Print "B is the largest number"
Print "C is the largest number"
Print "A is the largest number"
END

8. What is C programming language?List the advantages of it?

C is a powerful general-purpose programming language. It can be used to develop software like operating systems, databases, compilers, and so on.

Advantages:

As a middle-level language, C combines the features of both high-level and low-level languages. It can be used for low-level programming, such as scripting for drivers and kernels and it also supports functions of high-level programming languages, such as scripting for software applications etc.

C is a structured programming language which allows a complex program to be broken into simpler programs called functions. It also allows free movement of data across these functions.

Various features of C including direct access to machine level hardware APIs, the presence of C compilers, deterministic resource use and dynamic memory allocation make C language an optimum choice for scripting applications and drivers of embedded systems.

9. Define variable and mention the rules to be followed while creating a variable?

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location. For example:

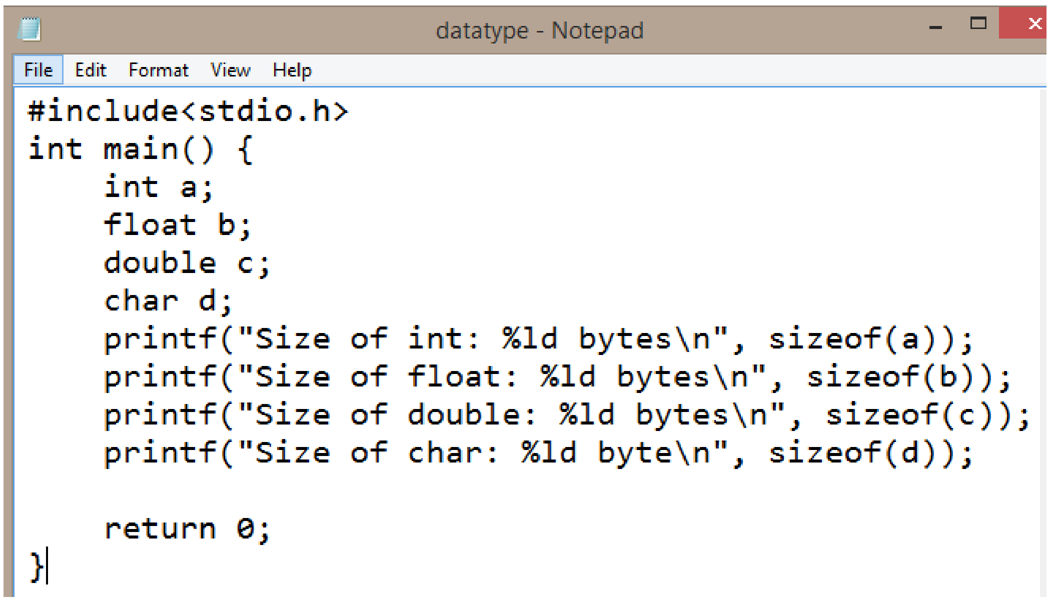
int playerScore = 95;

Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95.

Rules for declaring a variable:

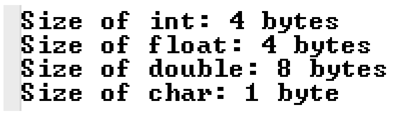
- Every variable name should start with alphabets or underscore (_).
- No spaces are allowed in variable declaration.
- Except underscore (_) no other special symbol are allowed in the middle of the variable declaration (not allowed -> roll-no, allowed -> roll_no).
- Maximum length of variable is 8 characters depend on compiler and operation system.
- Every variable name always should exist in the left hand side of assignment operator (invalid -> 10=a; valid -> a=10;).
- No keyword should access variable name (int for <- invalid because for is keyword).

10. Write a c program to find the sizeof each data type?



```c
#include<stdio.h>
int main() {
    int a;
    float b;
    double c;
    char d;
    printf("Size of int: %ld bytes\n", sizeof(a));
    printf("Size of float: %ld bytes\n", sizeof(b));
    printf("Size of double: %ld bytes\n", sizeof(c));
    printf("Size of char: %ld byte\n", sizeof(d));

    return 0;
}
```

Output:



```
Size of int: 4 bytes
Size of float: 4 bytes
Size of double: 8 bytes
Size of char: 1 byte
```

11. What is Operator?Mention the different kinds of operators in C?

An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

- Arithmetic operators
- Assignment operators
- Relational operators
- Logical operators
- Bit wise operators
- Conditional operators (ternary operators)
- Increment/decrement operators

- Special operators

12. What are Storage Classes?Mention their advantages?

storage class determines the scope, visibility and

lifetime of a variable.

There are 4 types of storage class:

• automatic

• external

• static

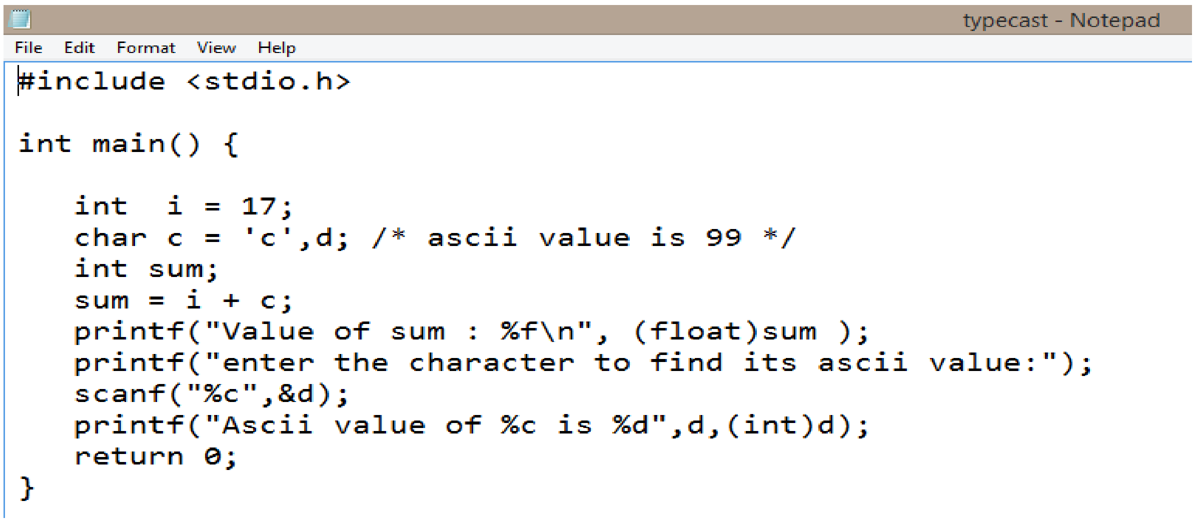
• register

Advantages:

- Where the variable is stored
- Scope of Variable
- Default initial value
- Lifetime of variable

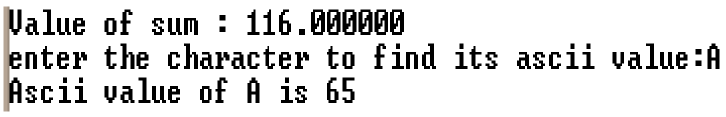
13. Write a c program to implement type conversion?

Program:



```c
#include <stdio.h>

int main() {

    int  i = 17;
    char c = 'c',d; /* ascii value is 99 */
    int sum;
    sum = i + c;
    printf("Value of sum : %f\n", (float)sum );
    printf("enter the character to find its ascii value:");
    scanf("%c",&d);
    printf("Ascii value of %c is %d",d,(int)d);
    return 0;
}
```

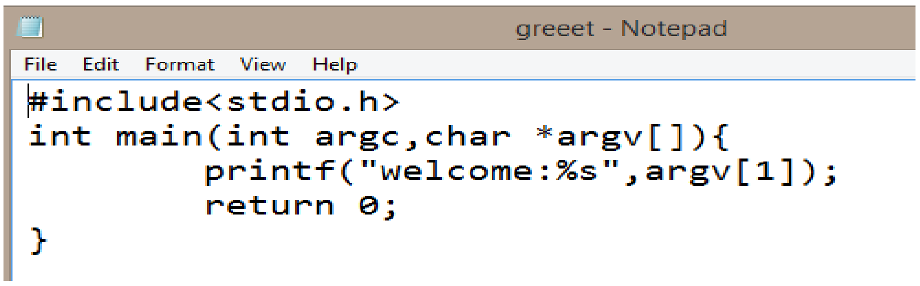
Output:



```
Value of sum : 116.000000
enter the character to find its ascii value:A
Ascii value of A is 65
```

14. Write a c program to accept command line arguments?

Program:

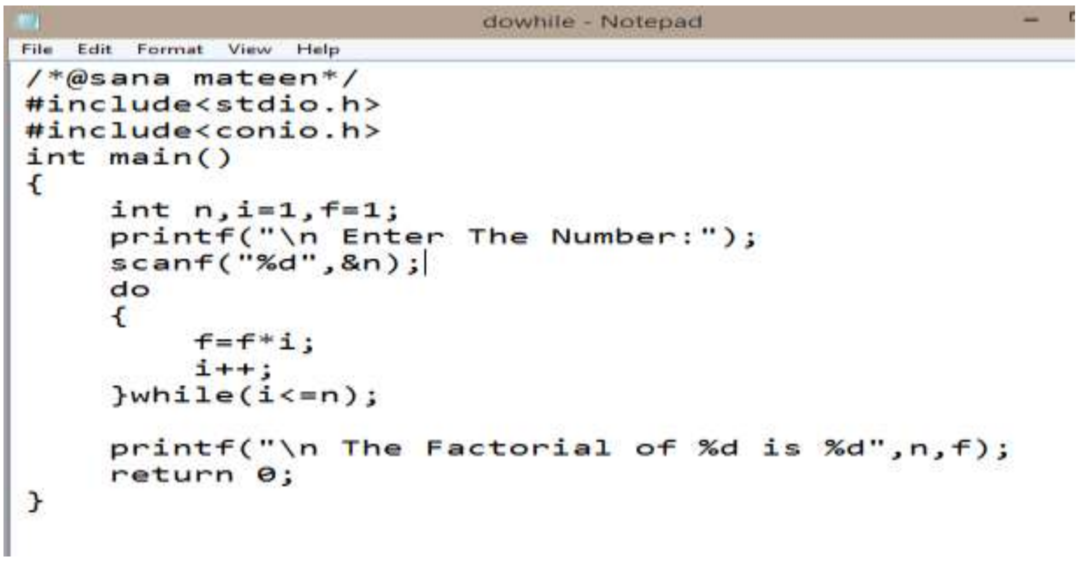


```c
#include<stdio.h>
int main(int argc,char *argv[]){
        printf("welcome:%s",argv[1]);
        return 0;
}
```

15. What are control statements?Write a c program to implement do while loop?

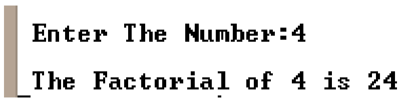
Control statements enable us to specify the flow of program control; ie, the order in which the instructions in a program must be executed.

Program:



```
/*@sana mateen*/
#include<stdio.h>
#include<conio.h>
int main()
{
	int n,i=1,f=1;
	printf("\n Enter The Number:");
	scanf("%d",&n);
	do
	{
		f=f*i;
		i++;
	}while(i<=n);

	printf("\n The Factorial of %d is %d",n,f);
	return 0;
}
```

Output:



```
Enter The Number:4
The Factorial of 4 is 24
```

16. Differentiate between while loop and do-while loop?

while loop	**do-while loop**
The while loop evaluates the test expression inside the parenthesis ().	The body of do...while loop is executed once. Only then, the test expression is evaluated.
If the test expression is true, statements	If the test expression is true, the body of the

inside the body of while loop are executed. Then, the test expression is evaluated again. The process goes on until the test expression is evaluated to false.	loop is executed again and the test expression is evaluated. This process goes on until the test expression becomes false.

If the test expression is false, the loop terminates (ends).
 	
while (testExpression)
 	
{
 	
// statements inside the body of the loop
 	
}
	If the test expression is false, the loop ends.
	
	do
	
	{
	
	// statements inside the body of the loop
	
	}
	
	while (testExpression);

17. What are command line arguments?

Command line argument is a parameter supplied to the program when it is invoked. Command line argument is an important concept in C programming. It is mostly used when you need to control your program from outside. Command line arguments are passed to the main() method.

Syntax:

int main(int argc, char *argv[])

Here argc counts the number of arguments on the command line and argv[ ] is a pointer array which holds pointers of type char which points to the arguments passed to the program.

Long Answer Questions for 5 Marks:

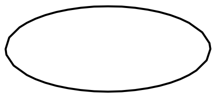
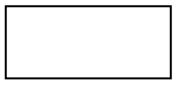
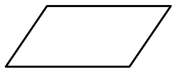
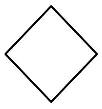
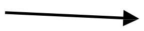
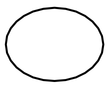
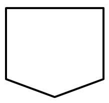
1. **What is flow chart? Explain the different notations used in flow chart?**

Flowchart is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

These are some points to keep in mind while developing a flowchart −

- Flowchart can have only one start and one stop symbol
- On-page connectors are referenced using numbers
- Off-page connectors are referenced using alphabets
- General flow of processes is top to bottom or left to right
- Arrows should not cross each other

Notations used:

Symbol	**Symbol Name**	**Purpose**
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

2. What is an Algorithm? Explain the characteristics of algorithm?

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

An algorithm should have the following characteristics −

Unambiguous − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.

Input − An algorithm should have 0 or more well-defined inputs.

Output − An algorithm should have 1 or more well-defined outputs, and should match the desired output.

Finiteness − Algorithms must terminate after a finite number of steps.

Feasibility − Should be feasible with the available resources.

Independent − An algorithm should have step-by-step directions, which should be independent of any programming code.

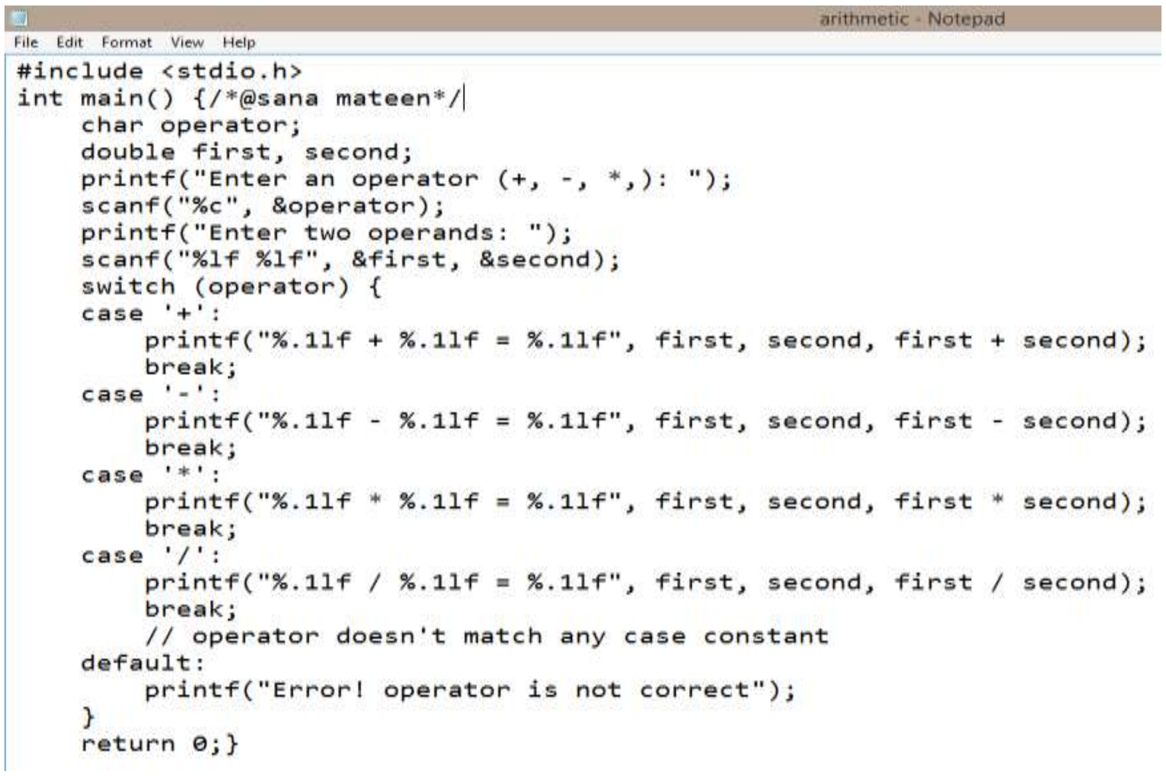
Example:

Design an algorithm to add two numbers and display the result.

```
Step 1 − START

Step 2 − declare three integers a, b & c

Step 3 − define values of a & b

Step 4 − add values of a & b

Step 5 − store output of step 4 to c

Step 6 − print c

Step 7 − STOP
```

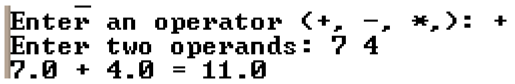
3. **Write a c program to perform arithmetic operations using switch case?**

Program:



```
#include <stdio.h>
int main() {/*@sana mateen*/
    char operator;
    double first, second;
    printf("Enter an operator (+, -, *,): ");
    scanf("%c", &operator);
    printf("Enter two operands: ");
    scanf("%lf %lf", &first, &second);
    switch (operator) {
    case '+':
        printf("%.1lf + %.1lf = %.1lf", first, second, first + second);
        break;
    case '-':
        printf("%.1lf - %.1lf = %.1lf", first, second, first - second);
        break;
    case '*':
        printf("%.1lf * %.1lf = %.1lf", first, second, first * second);
        break;
    case '/':
        printf("%.1lf / %.1lf = %.1lf", first, second, first / second);
        break;
        // operator doesn't match any case constant
    default:
        printf("Error! operator is not correct");
    }
    return 0;}
```

Output:



```
Enter an operator (+, -, *,): +
Enter two operands: 7 4
7.0 + 4.0 = 11.0
```

4. **Explain the general structure of c program along with an example?**

Basic Structure of C Program

Documentation Section

This section consists of comment lines which include the name of programmer, the author and other details like time and date of writing the program. Documentation section helps anyone to get an overview of the program.

Link Section

The link section consists of the header files of the functions that are used in the program. It provides instructions to the compiler to link functions from the system library.

Definition Section

All the symbolic constants are written in definition section. Macros are known as symbolic constants.

Global Declaration Section

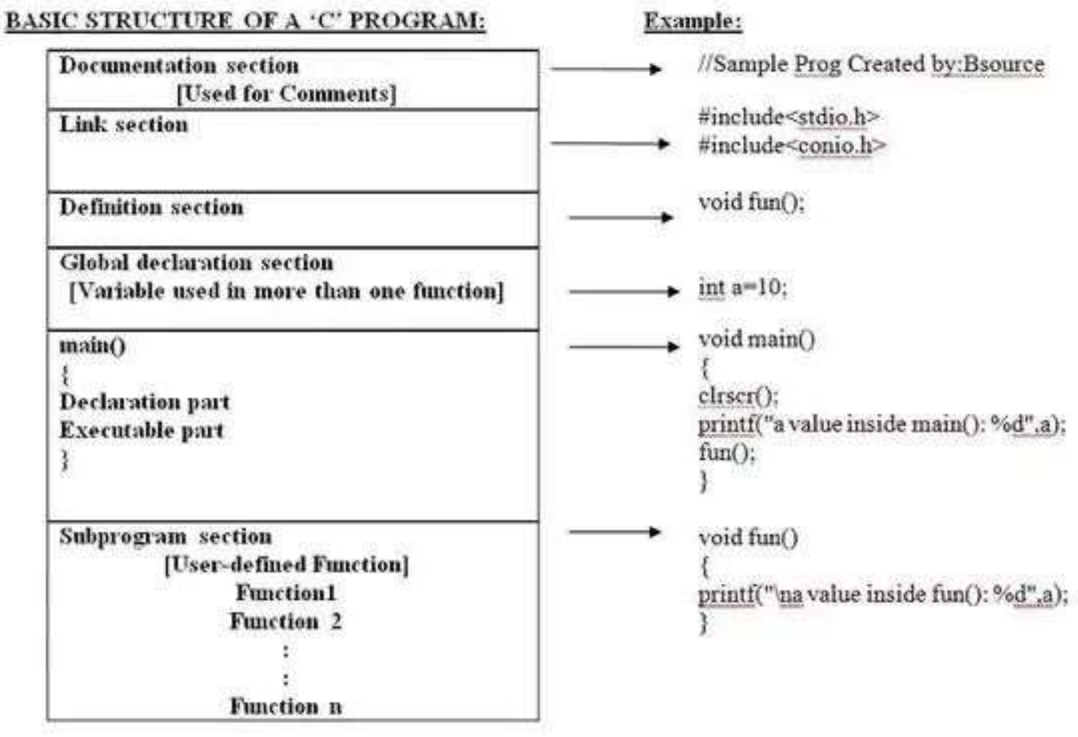
The global variables that can be used anywhere in the program are declared in global declaration section. This section also declares the user defined functions.

main() Function Section

It is necessary have one main() function section in every C program. This section contains two parts, declaration and executable part. The declaration part declares all the variables that are used in executable part. These two parts must be written in between the opening and closing braces. Each statement in the declaration and executable part must end with a semicolon (;). The execution of program starts at opening braces and ends at closing braces.

Subprogram Section

The subprogram section contains all the user defined functions that are used to perform a specific task. These user defined functions are called in the main() function.


BASIC STRUCTURE OF A 'C' PROGRAM:
Example:
Documentation section [Used for Comments]
Link section
Definition section
Global declaration section [Variable used in more than one function]
main() { Declaration part Executable part }
Subprogram section [User-defined Function] Function1 Function 2 : : Function n

```
//Sample Prog Created by:Bsource
#include<stdio.h>
#include<conio.h>
void fun();
int a=10;
void main()
{
clrscr();
printf("a value inside main(): %d",a);
fun();
}
void fun()
{
printf("\na value inside fun(): %d",a);
}
```

5. **Write a c program to implement ternary operator?**

In C Programming, ternary operator allows executing different code depending on the value of a condition. The returned value is the result of the expression when the code is executed. The main advantage of using ternary operator is to reduce the number of lines of code and improve the performance of application.

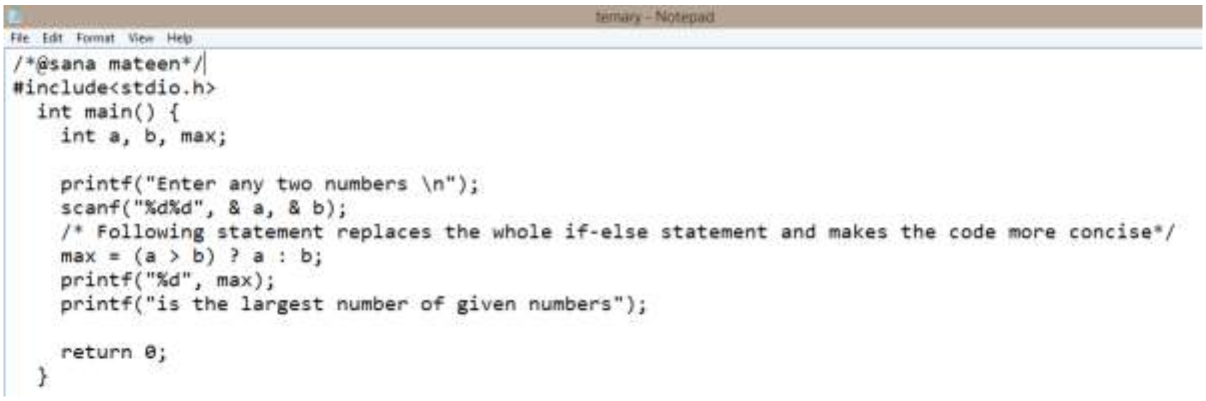
In C, the real utility of ternary operator is that it is an expression instead of a statement i.e. you can have it on the right-hand side (RHS) of a statement. So you can write certain code statements more concisely.

Expression1 ? Expression2 : Expression3

OR

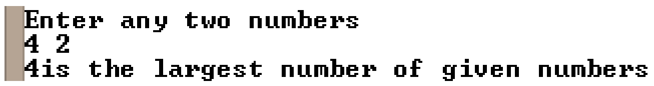
(condition) ? (if_true) : (if_false)

Program:



```
/*@sana mateen*/
#include<stdio.h>
  int main() {
    int a, b, max;

    printf("Enter any two numbers \n");
    scanf("%d%d", & a, & b);
    /* Following statement replaces the whole if-else statement and makes the code more concise*/
    max = (a > b) ? a : b;
    printf("%d", max);
    printf("is the largest number of given numbers");

    return 0;
  }
```

Output:



```
Enter any two numbers
4 2
4is the largest number of given numbers
```

6. **What is an operator? Explain different types of operator along with an example?**

An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

C has a wide range of operators to perform various operations.

C Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

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

C Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

C Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =

Operator	Example	Same as
=	**a = b**	**a = b**
+=	**a += b**	**a = a+b**
-=	**a -= b**	**a = a-b**
*=	**a *= b**	**a = a*b**
/=	**a /= b**	**a = a/b**
%=	**a %= b**	**a = a%b**

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	**Meaning of Operator**	**Example**
==	**Equal to**	**5 == 3 is evaluated to 0**
>	**Greater than**	**5 > 3 is evaluated to 1**
<	**Less than**	**5 < 3 is evaluated to 0**
!=	**Not equal to**	**5 != 3 is evaluated to 1**
>=	**Greater than or equal to**	**5 >= 3 is evaluated to 1**
<=	**Less than or equal to**	**5 <= 3 is evaluated to 0**

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	**Meaning**	**Example**
&&	**Logical AND. True only if all operands are true**	**If c = 5 and d = 2 then, expression ((c==5) && (d>5)) equals to 0.**
||	**Logical OR. True only if either one operand is true**	**If c = 5 and d = 2 then, expression ((c==5) || (d>5)) equals to 1.**
!	**Logical NOT. True only if the operand is 0**	**If c = 5 then, expression !(c==5) equals to 0.**

C Bitwise Operators

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power.

Bitwise operators are used in C programming to perform bit-level operations.

Operators	**Meaning of operators**

&	**Bitwise AND**
|	**Bitwise OR**
^	**Bitwise exclusive OR**
~	**Bitwise complement**
<<	**Shift left**
>>	**Shift right**

Comma Operator

Comma operators are used to link related expressions together. For example:

int a, c = 5, d;

The sizeof operator

The sizeof is a unary operator that returns the size of data (constants, variables, array, structure, etc).

7. Explain operator precedence in detail?

Precedence of operators

The precedence of operators determines which operator is executed first if there is more than one operator in an expression.

Let us consider an example:

int x = 5 - 17* 6;

In C, the precedence of * is higher than - and =. Hence, 17 * 6 is evaluated first. Then the expression involving - is evaluated as the precedence of - is higher than that of =.

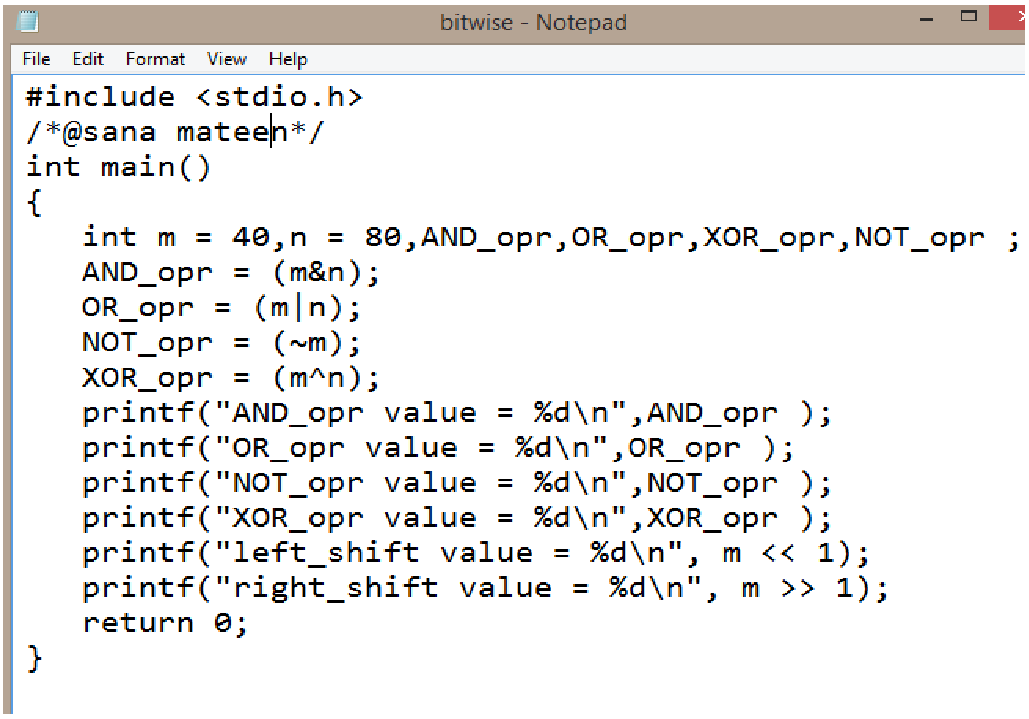
Here's a table of operators precedence from higher to lower. The property of associativity will be discussed shortly.

Operator	Meaning of operator	Associativity
() [] -> .	Functional call Array element reference Indirect member selection Direct member selection	Left to right
! ~ + -	Logical negation Bitwise(1 's) complement Unary plus Unary minus	Right to left

++ -- & * sizeof (type)	Increment Decrement Dereference (Address) Pointer reference Returns the size of an object Typecast (conversion)	
* / %	Multiply Divide Remainder	Left to right
+ -	Binary plus(Addition) Binary minus(subtraction)	Left to right
<< >>	Left shift Right shift	Left to right
< <= > >=	Less than Less than or equal Greater than Greater than or equal	Left to right
== !=	Equal to Not equal to	Left to right
&	Bitwise AND	Left to right
^	Bitwise exclusive OR	Left to right
|	Bitwise OR	Left to right
&&	Logical AND	Left to right
||	Logical OR	Left to right
?:	Conditional Operator	Right to left
= *= /= %= += -= &= ^= |= <<= >>=	Simple assignment Assign product Assign quotient Assign remainder Assign sum Assign difference Assign bitwise AND Assign bitwise XOR Assign bitwise OR Assign left shift Assign right shift	Right to left
,	Separator of expressions	Left to right

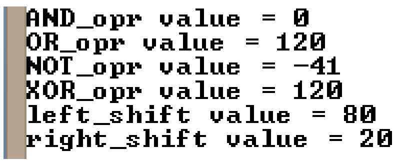
8. Write a c program to implement bitwise operator ?

Program:



```
#include <stdio.h>
/*@sana mateen*/
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);
   return 0;
}
```

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
```

9. Explain about storage classes in detail along with an example program?

Every variable in C programming has two properties: type and storage class.

Type refers to the data type of a variable. And, storage class determines the scope, visibility and lifetime of a variable.

There are 4 types of storage class:

automatic

external

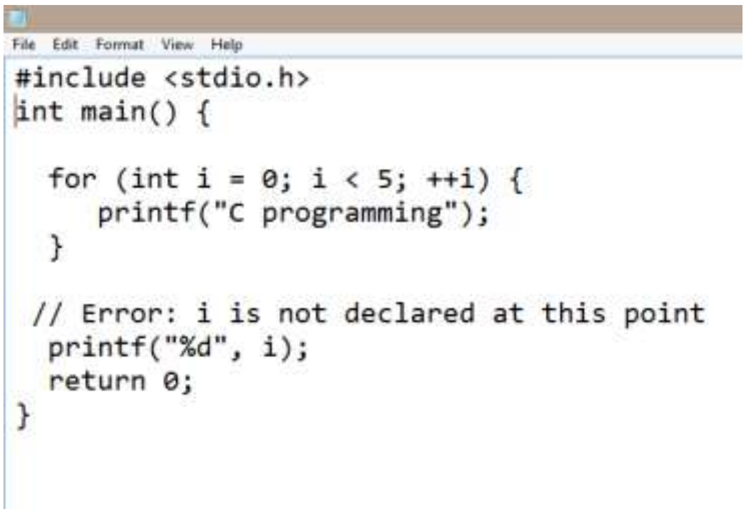
static

register

Local Variable

The variables declared inside a block are automatic or local variables. The local variables exist only inside the block in which it is declared.

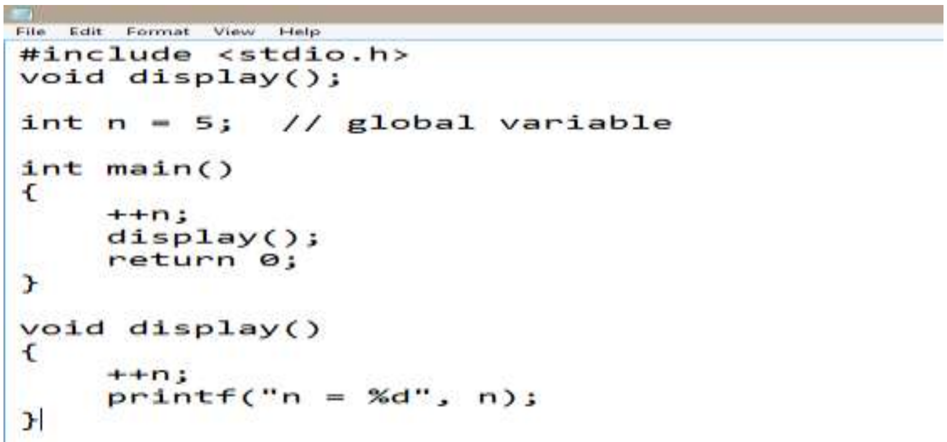
Let's take an example.



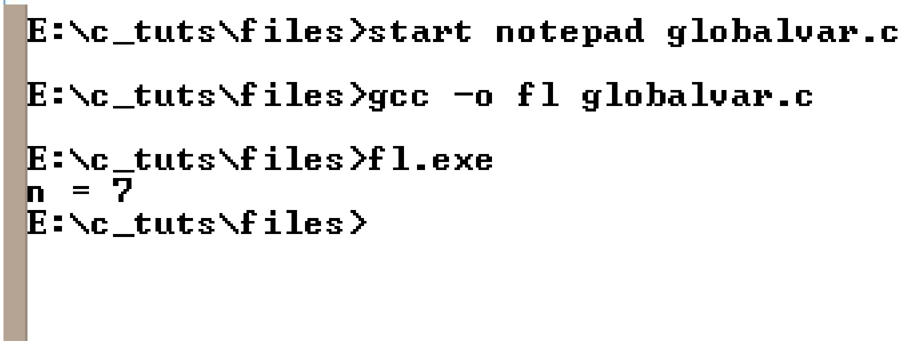
```
#include <stdio.h>
int main() {

  for (int i = 0; i < 5; ++i) {
     printf("C programming");
  }

 // Error: i is not declared at this point
  printf("%d", i);
  return 0;
}
```

Global Variable:

Variables that are declared outside of all functions are known as external or global variables. They are accessible from any function inside the program.



```
#include <stdio.h>
void display();

int n = 5;   // global variable

int main()
{
      ++n;
      display();
      return 0;
}

void display()
{
      ++n;
      printf("n = %d", n);
}
```



```
E:\c_tuts\files>start notepad globalvar.c

E:\c_tuts\files>gcc -o f1 globalvar.c

E:\c_tuts\files>f1.exe
n = 7
E:\c_tuts\files>
```

Register Variable

The register keyword is used to declare register variables. Register variables were supposed to be faster than local variables.

However, modern compilers are very good at code optimization, and there is a rare chance that using register variables will make your program faster.

Unless you are working on embedded systems where you know how to optimize code for the given application, there is no use of register variables.

Static Variable

A static variable is declared by using the static keyword. For example;

static int i;

The value of a static variable persists until the end of the program.

10. Explain conditional statements in c along with an example?

Conditional statements help you to make a decision based on certain conditions. These conditions are specified by a set of conditional statements having boolean expressions which are evaluated to a boolean value true or false. There are following types of conditional statements in C.

- If statement
- If-Else statement
- Nested If-else statement
- If-Else If ladder
- Switch statement

If statement

The single if statement in C language is used to execute the code if a condition is true. It is also called one-way selection statement.

Syntax

```
if(expression)
{
 //code to be executed
}
```

How "if" statement works..

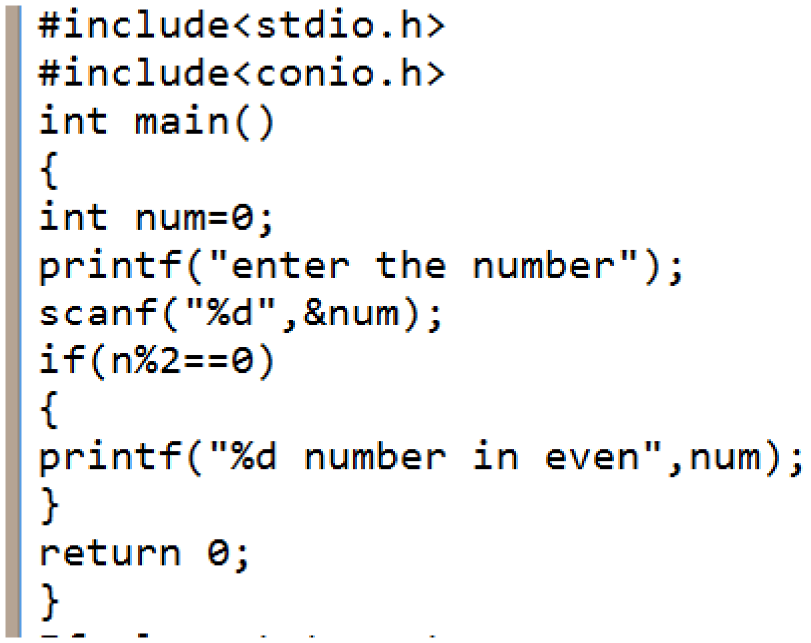
If the expression is evaluated to nonzero (true) then if block statement(s) are executed.

If the expression is evaluated to zero (false) then Control passes to the next statement following it.

Note

"Expression must be scalar type" i.e evaluated to a single value.

If-else statement



```
#include<stdio.h>
#include<conio.h>
int main()
{
int num=0;
printf("enter the number");
scanf("%d",&num);
if(n%2==0)
{
printf("%d number in even",num);
}
return 0;
}
```

The if-else statement in C language is used to execute the code if condition is true or false. It is also called two-way selection statement.

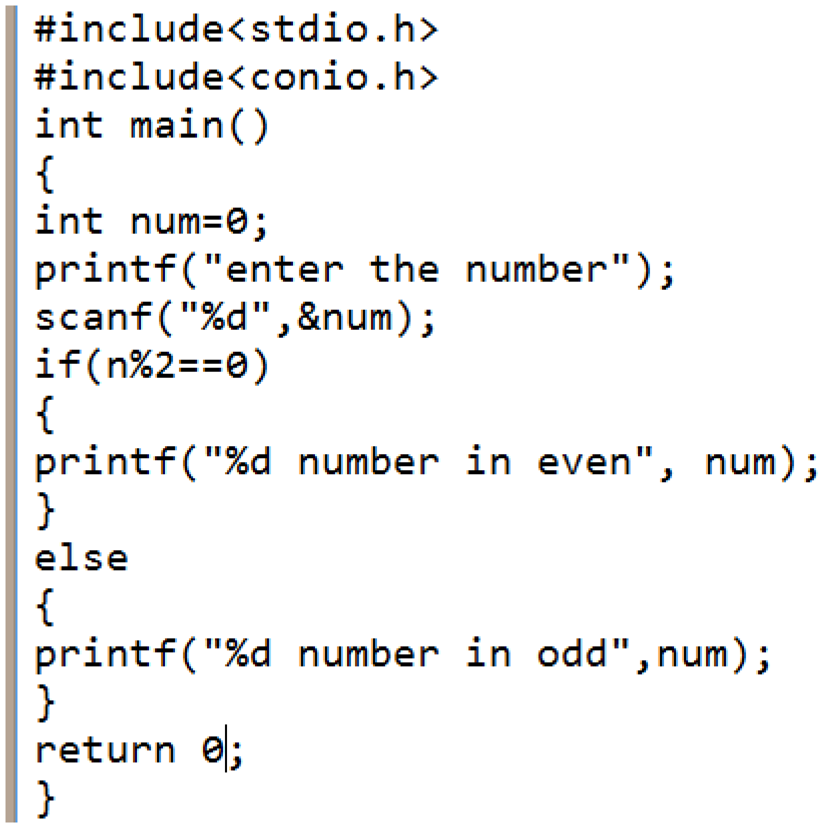
Syntax

```
if(expression)

{

 //Statements

}

else

{

 //Statements

}
```

How "if..else" statement works..

If the expression is evaluated to nonzero (true) then if block statement(s) are executed.

If the expression is evaluated to zero (false) then else block statement(s) are executed.



```
#include<stdio.h>
#include<conio.h>
int main()
{
int num=0;
printf("enter the number");
scanf("%d",&num);
if(n%2==0)
{
printf("%d number in even", num);
}
else
{
printf("%d number in odd",num);
}
return 0;
}
```

Nested If-else statement

The nested if...else statement is used when a program requires more than one test expression. It is also called a multi-way selection statement. When a series of the decision are involved in a statement, we use if else statement in nested form.

Syntax

```
if( expression )
{
 if( expression1 )
 {
 statement-block1;
 }
 else
 {
 statement-block 2;
```

}

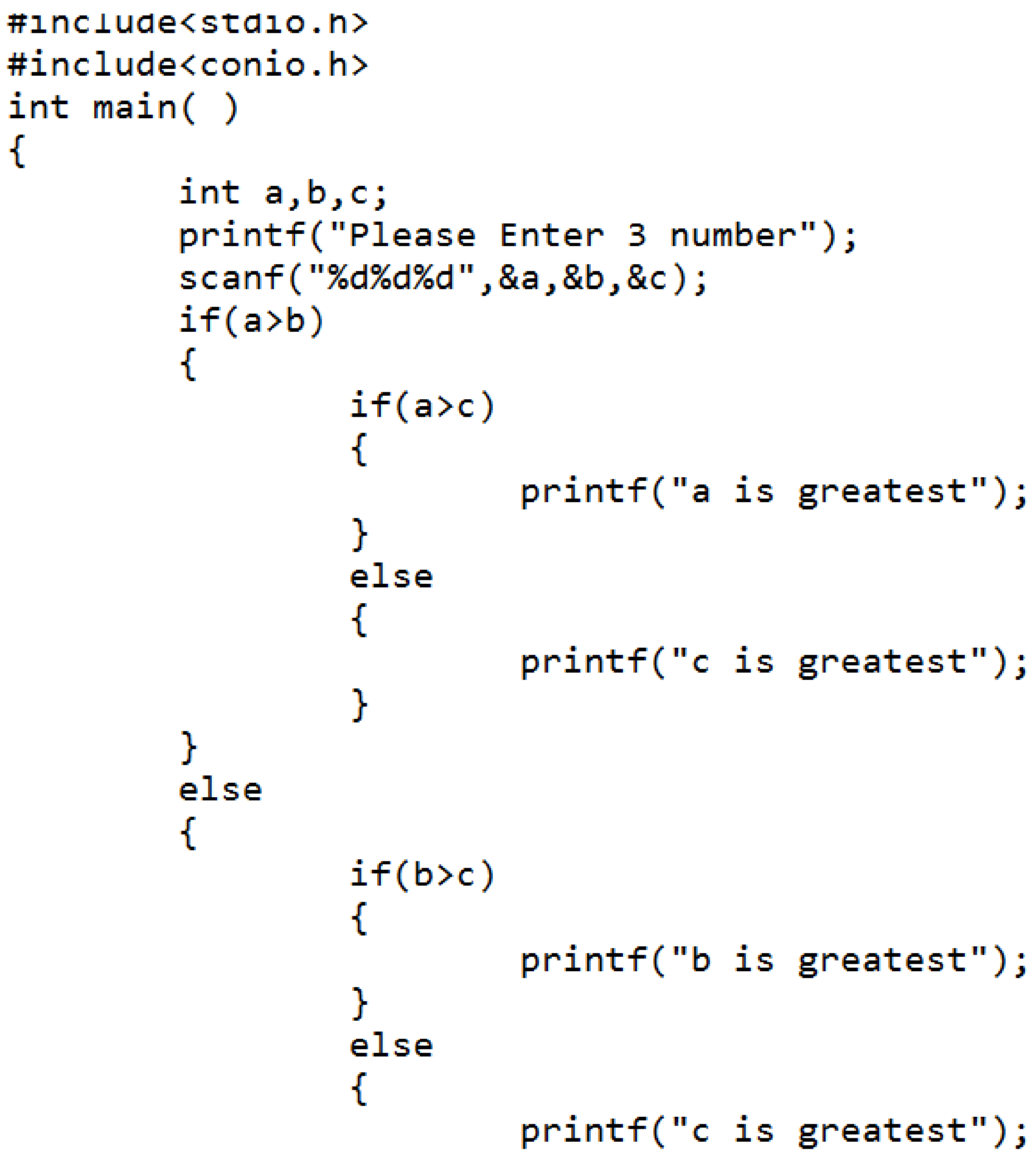
}

else

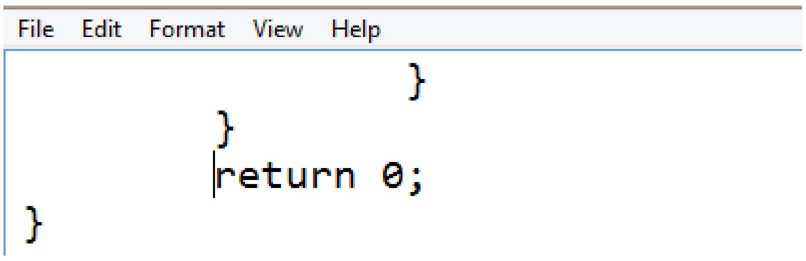
{

statement-block 3;

}



```
#include<stdio.h>
#include<conio.h>
int main( )
{
        int a,b,c;
        printf("Please Enter 3 number");
        scanf("%d%d%d",&a,&b,&c);
        if(a>b)
        {
                if(a>c)
                {
                        printf("a is greatest");
                }
                else
                {
                        printf("c is greatest");
                }
        }
        else
        {
                if(b>c)
                {
                        printf("b is greatest");
                }
                else
                {
                        printf("c is greatest");
```



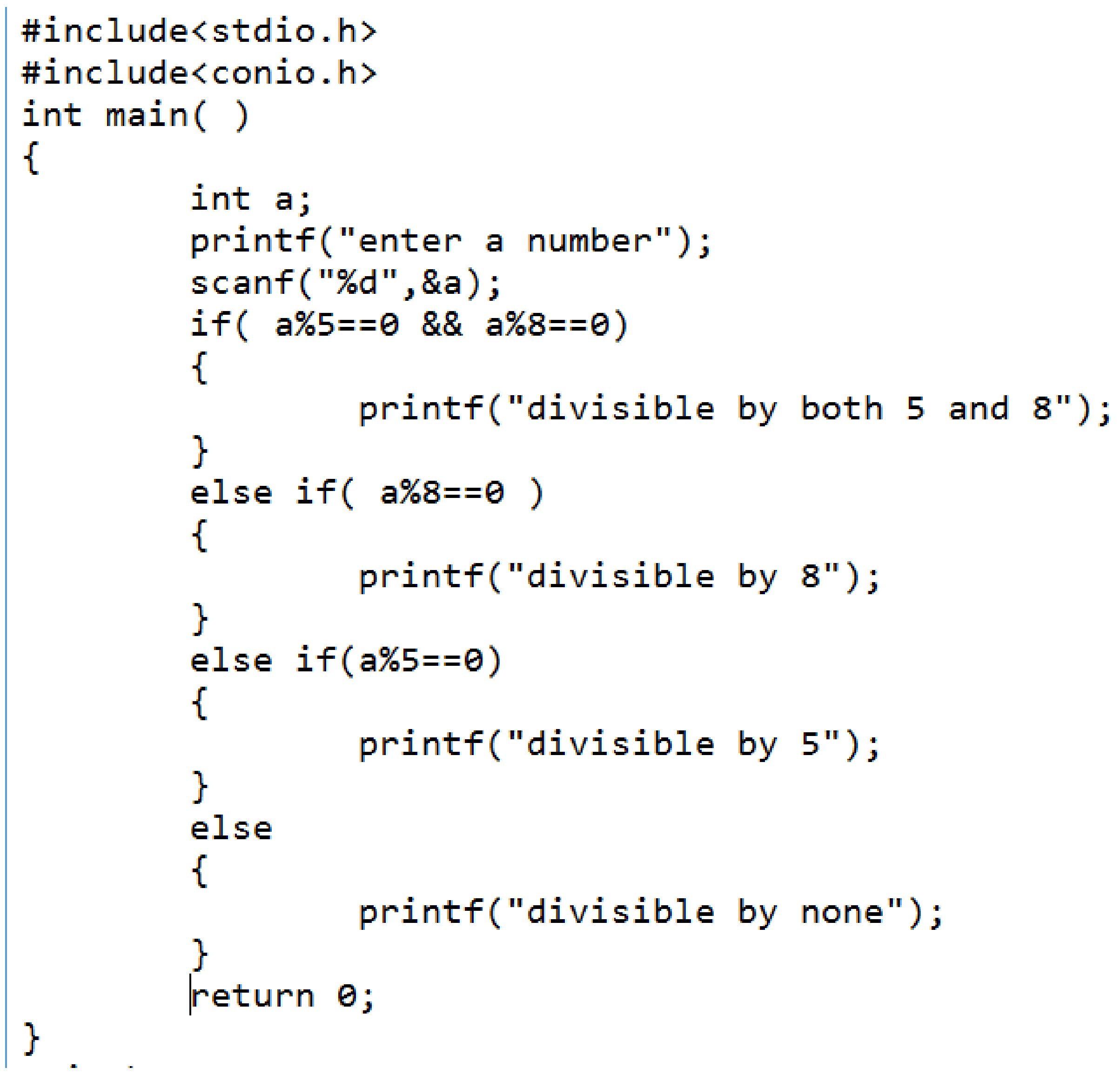
```
                        }
            }
            return 0;
}
```

If..else If ladder

The if-else-if statement is used to execute one code from multiple conditions. It is also called multipath decision statement. It is a chain of if..else statements in which each if statement is associated with else if statement and last would be an else statement.

Syntax

```
if(condition1)
{
 //statements
}
else if(condition2)
{
 //statements
}
else if(condition3)
{
 //statements
}
else
{
 //statements
}
```



```
#include<stdio.h>
#include<conio.h>
int main( )
{
        int a;
        printf("enter a number");
        scanf("%d",&a);
        if( a%5==0 && a%8==0)
        {
                printf("divisible by both 5 and 8");
        }
        else if( a%8==0 )
        {
                printf("divisible by 8");
        }
        else if(a%5==0)
        {
                printf("divisible by 5");
        }
        else
        {
                printf("divisible by none");
        }
        return 0;
}
```

Switch Statement

switch statement acts as a substitute for a long if-else-if ladder that is used to test a list of cases. A switch statement contains one or more case labels which are tested against the switch expression. When the expression match to a case then the associated statements with that case would be executed.

Syntax

switch (expression)

{

 case value1:

 //Statements

break;

case value 2:

//Statements

break;

case value 3:

//Statements

case value n:

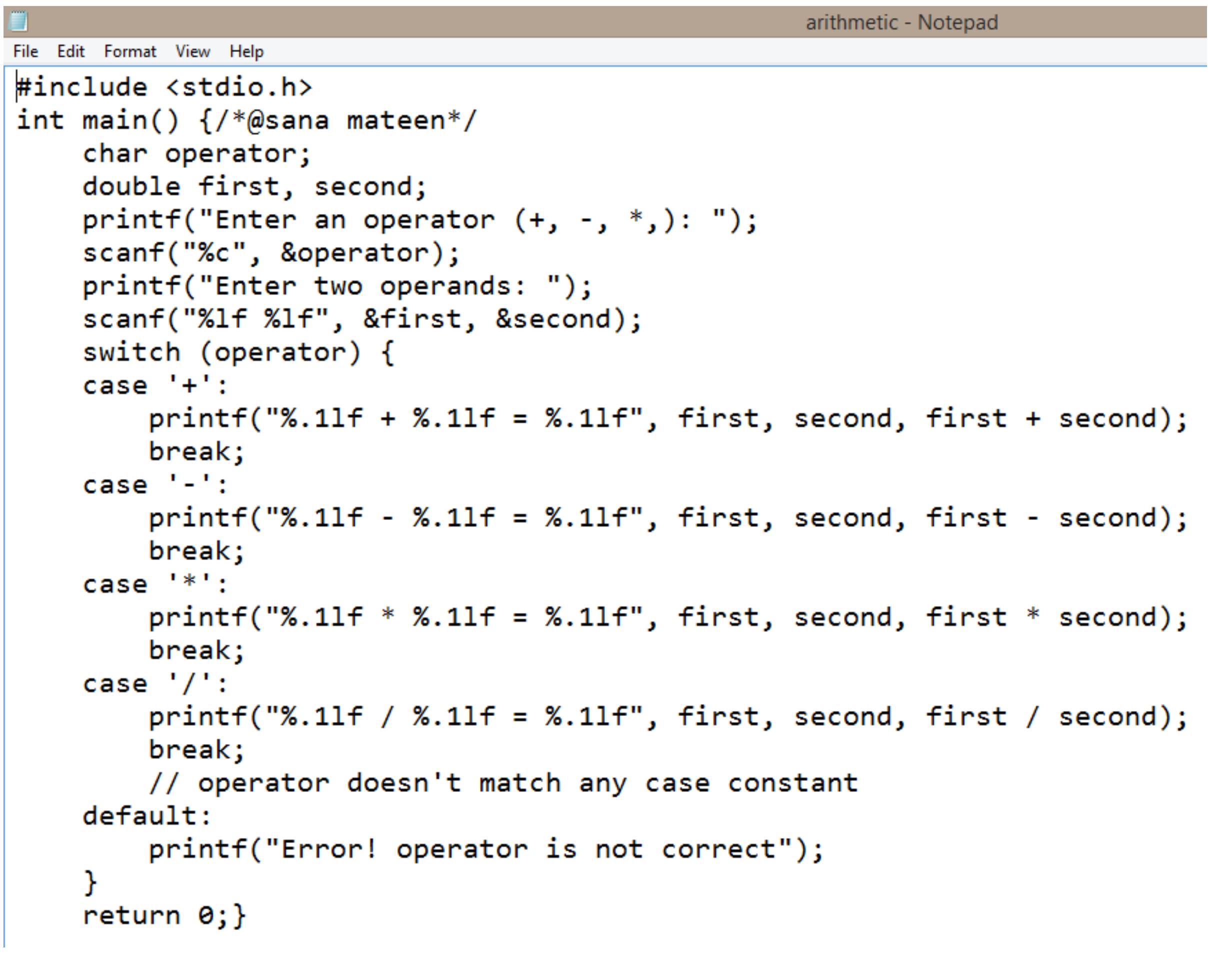
//Statements

break;

default:

//Statements

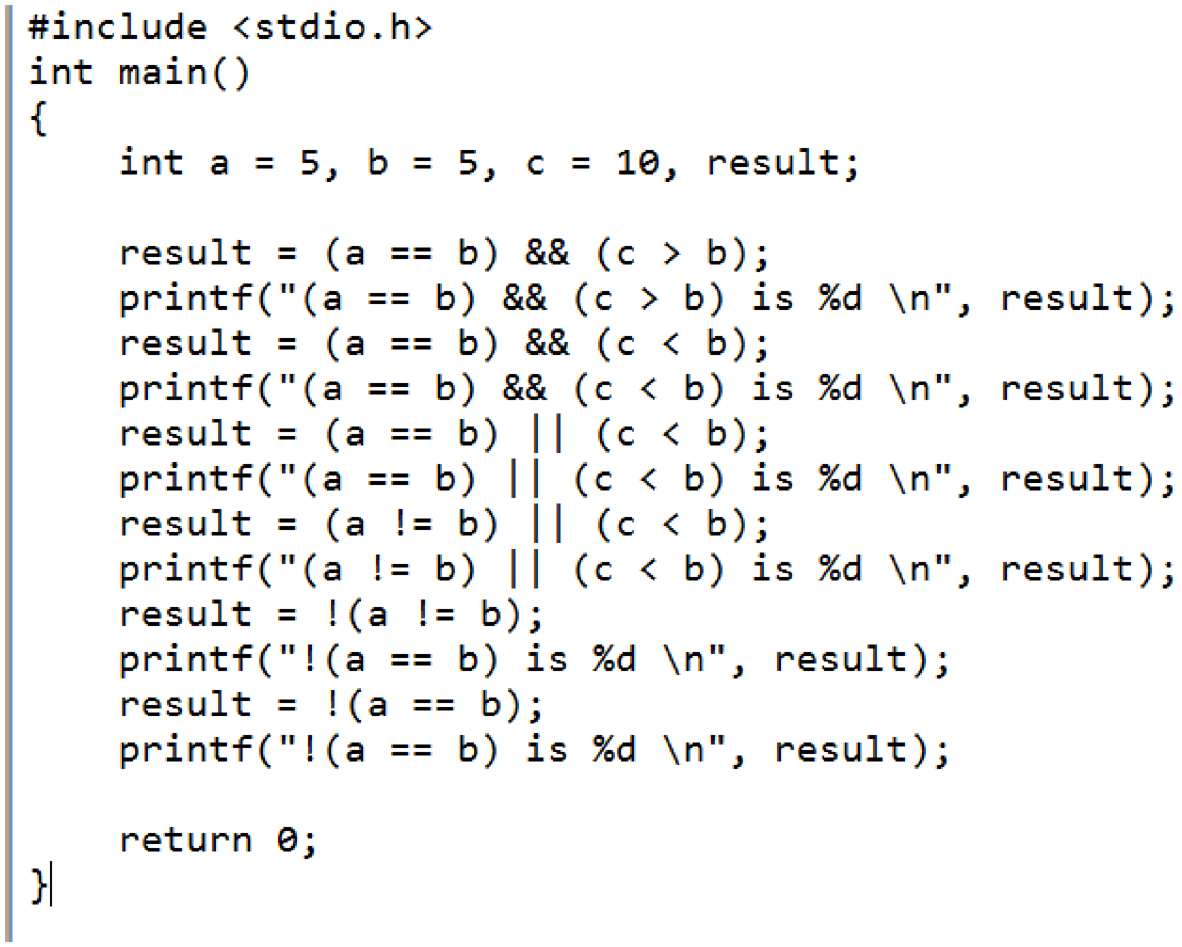
}



```
#include <stdio.h>
int main() {/*@sana mateen*/
    char operator;
    double first, second;
    printf("Enter an operator (+, -, *,): ");
    scanf("%c", &operator);
    printf("Enter two operands: ");
    scanf("%lf %lf", &first, &second);
    switch (operator) {
    case '+':
        printf("%.1lf + %.1lf = %.1lf", first, second, first + second);
        break;
    case '-':
        printf("%.1lf - %.1lf = %.1lf", first, second, first - second);
        break;
    case '*':
        printf("%.1lf * %.1lf = %.1lf", first, second, first * second);
        break;
    case '/':
        printf("%.1lf / %.1lf = %.1lf", first, second, first / second);
        break;
        // operator doesn't match any case constant
    default:
        printf("Error! operator is not correct");
    }
    return 0;}
```

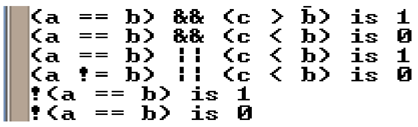
11. Write a c program to implement logical operators ?

Program:



```c
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);
    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) is %d \n", result);
    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) is %d \n", result);
    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) is %d \n", result);
    result = !(a != b);
    printf("!(a == b) is %d \n", result);
    result = !(a == b);
    printf("!(a == b) is %d \n", result);

    return 0;
}
```

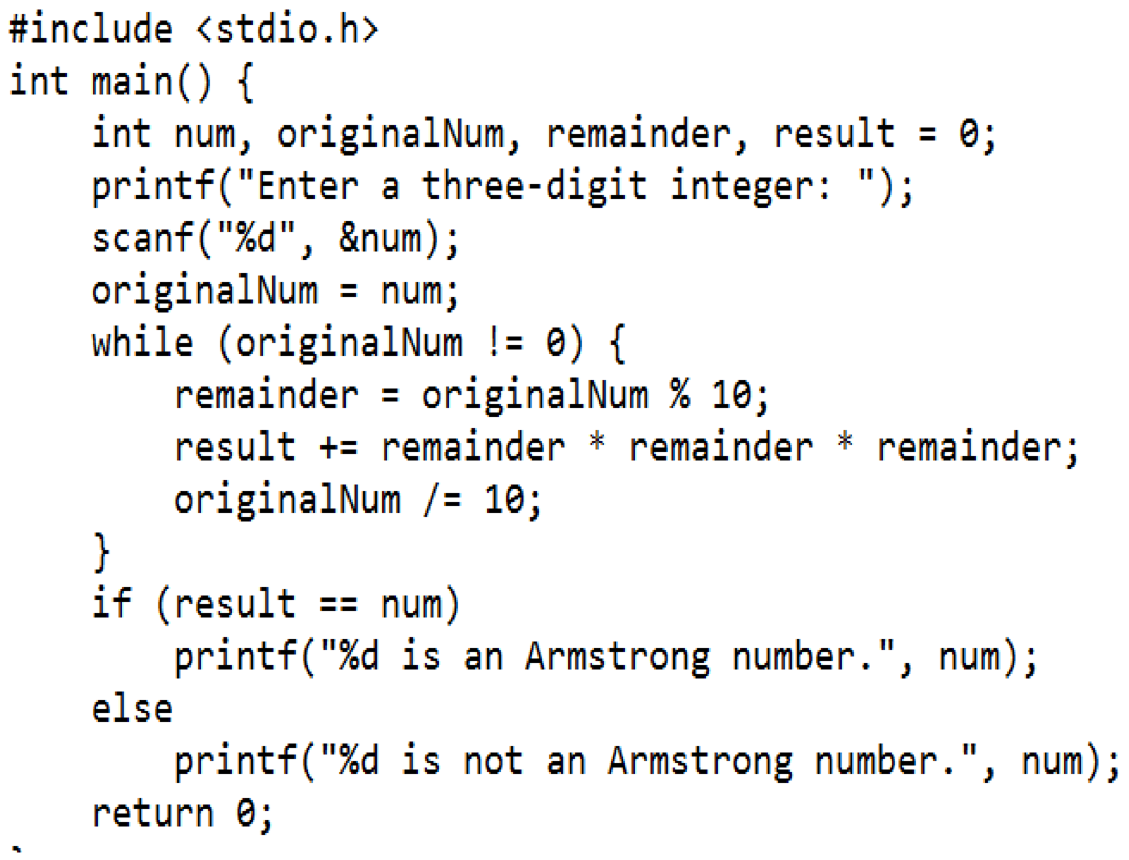
Output:



```
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a == b) is 1
!(a == b) is 0
```

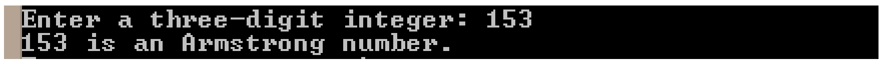
12. Write a c program to find Armstrong number?

Program:



```
#include <stdio.h>
int main() {
    int num, originalNum, remainder, result = 0;
    printf("Enter a three-digit integer: ");
    scanf("%d", &num);
    originalNum = num;
    while (originalNum != 0) {
        remainder = originalNum % 10;
        result += remainder * remainder * remainder;
        originalNum /= 10;
    }
    if (result == num)
        printf("%d is an Armstrong number.", num);
    else
        printf("%d is not an Armstrong number.", num);
    return 0;
```

Output:



```
Enter a three-digit integer: 153
153 is an Armstrong number.
```

13. what are command line arguments? explain with an example program?

main() function of a C program accepts arguments from command line or from other shell scripts by following commands. They are,

argc

argv[]

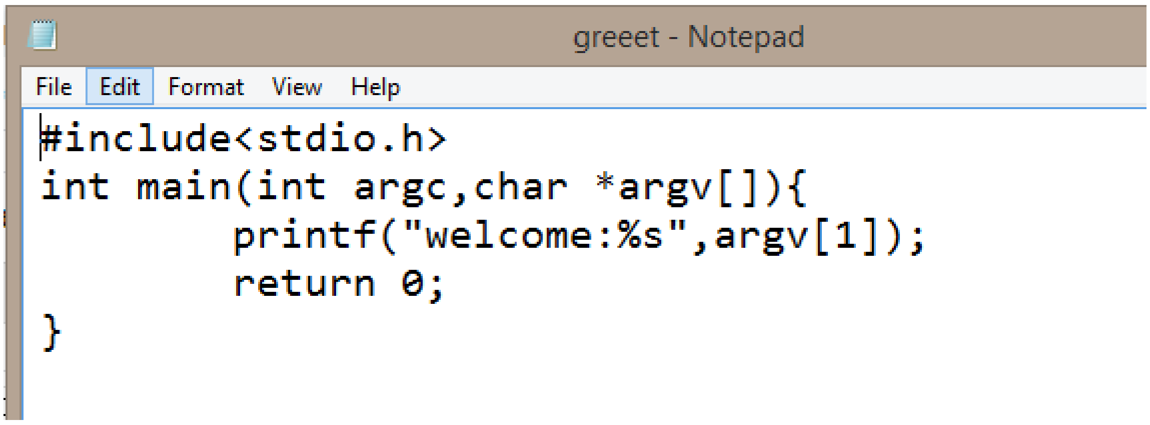
where,

argc – Number of arguments in the command line including program name argv[] – This is carrying all the arguments

In real time application, it will happen to pass arguments to the main program itself. These arguments are passed to the main () function while executing binary file from command line.

For example, when we compile a program (test.c), we get executable file in the name "test".

Now, we run the executable "test" along with 4 arguments in command line like below.



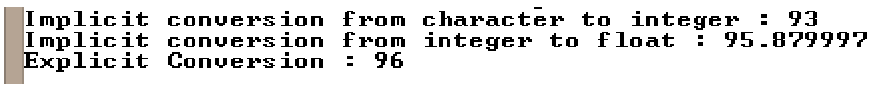
```
#include<stdio.h>
int main(int argc,char *argv[]){
        printf("welcome:%s",argv[1]);
        return 0;
}
```

14. Write a c program to implement implicit and explicit type conversion?

Program:

```
#include <stdio.h>
int main() {
   int a = 10;
   char b = 'S';
   float c = 2.88;
   a = a+b;
   printf("Implicit conversion from character to integer : %d\n",a);
   c = c+a;
   printf("Implicit conversion from integer to float : %f\n",c);
   int s = (int)c+1;
   printf("Explicit Conversion : %d\n",s);
   return 0;
}
```

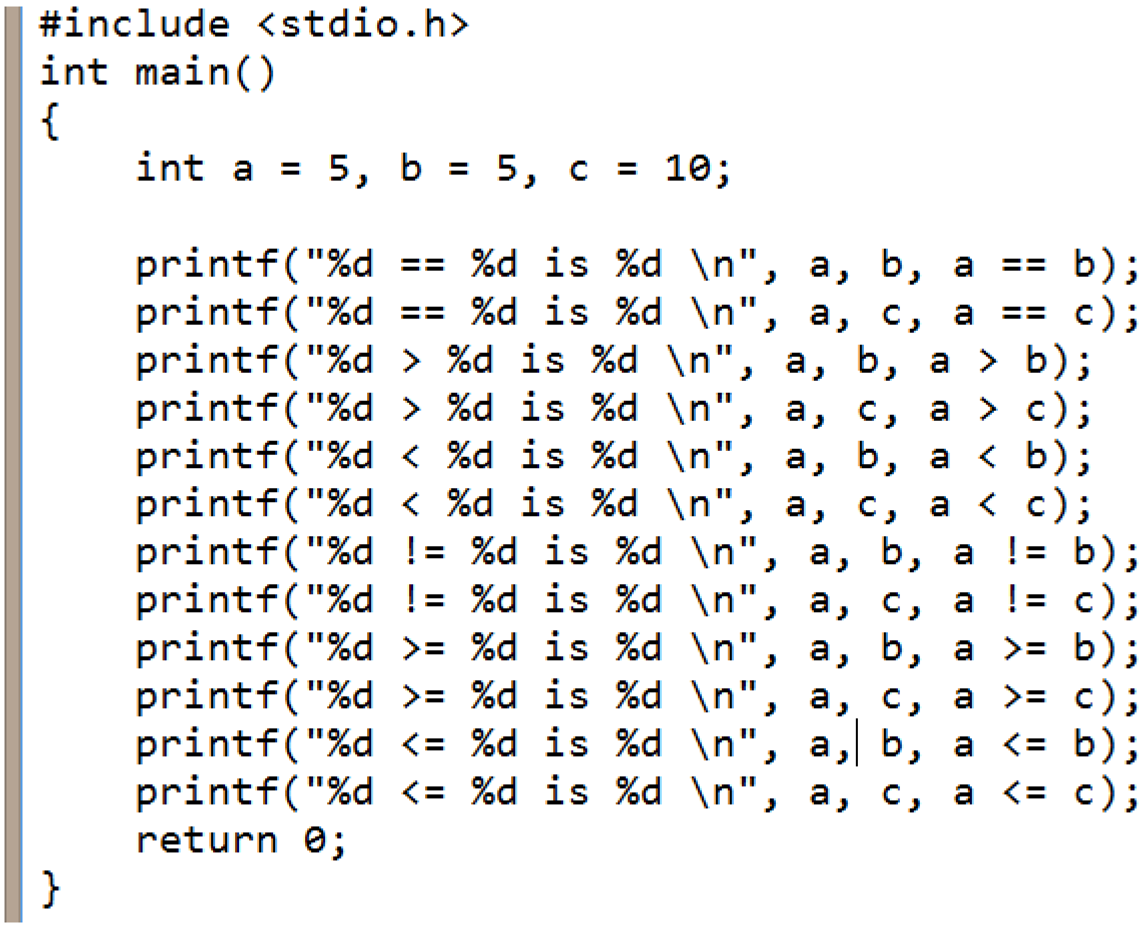
Output:



```
Implicit conversion from character to integer : 93
Implicit conversion from integer to float : 95.879997
Explicit Conversion : 96
```

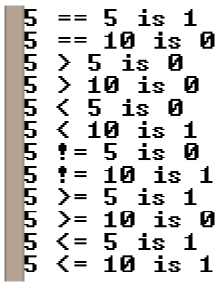
15. Write a c program to implement relational operators?

Program:



```c
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;

    printf("%d == %d is %d \n", a, b, a == b);
    printf("%d == %d is %d \n", a, c, a == c);
    printf("%d > %d is %d \n", a, b, a > b);
    printf("%d > %d is %d \n", a, c, a > c);
    printf("%d < %d is %d \n", a, b, a < b);
    printf("%d < %d is %d \n", a, c, a < c);
    printf("%d != %d is %d \n", a, b, a != b);
    printf("%d != %d is %d \n", a, c, a != c);
    printf("%d >= %d is %d \n", a, b, a >= b);
    printf("%d >= %d is %d \n", a, c, a >= c);
    printf("%d <= %d is %d \n", a, b, a <= b);
    printf("%d <= %d is %d \n", a, c, a <= c);
    return 0;
}
```

Output:



```
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
5 >= 10 is 0
5 <= 5 is 1
5 <= 10 is 1
```

16. Explain the process of accepting input and output through scanf and printf?

stdio.h header file is required for input and output operations in C. In this chapter we will discuss two input functions: scanf() and getchar() and two output functions: printf() and putchar(). But first, we will study something called conversion specification because functions like scanf() and printf() use this facility.

PRINTF() FUNCTION IN C LANGUAGE:

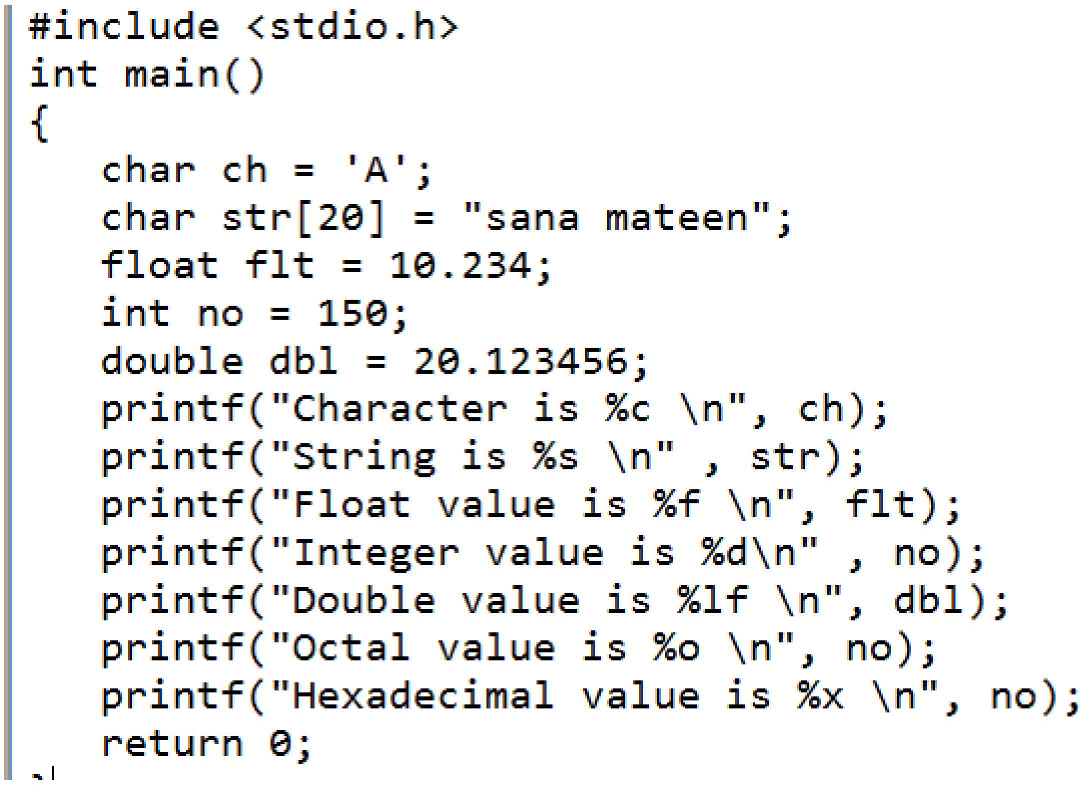
In C programming language, printf() function is used to print the “character, string, float, integer, octal and hexadecimal values” onto the output screen.

We use printf() function with %d format specifier to display the value of an integer variable.

Similarly %c is used to display character, %f for float variable, %s for string variable, %lf for double and %x for hexadecimal variable.

To generate a newline,we use “\n” in C printf() statement.

C language is case sensitive. For example, printf() and scanf() are different from Printf() and Scanf(). All characters in printf() and scanf() functions must be in lower case.



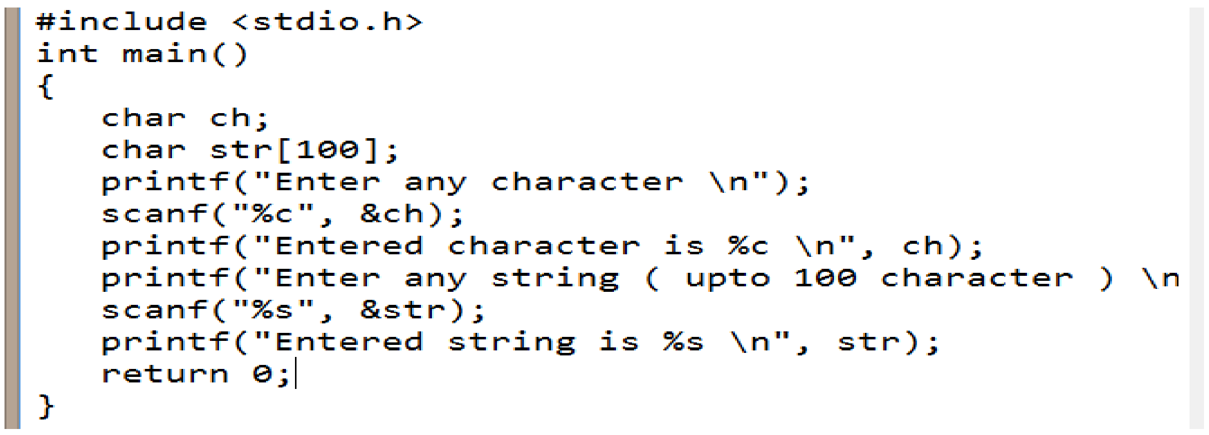
```
#include <stdio.h>
int main()
{
   char ch = 'A';
   char str[20] = "sana mateen";
   float flt = 10.234;
   int no = 150;
   double dbl = 20.123456;
   printf("Character is %c \n", ch);
   printf("String is %s \n" , str);
   printf("Float value is %f \n", flt);
   printf("Integer value is %d\n" , no);
   printf("Double value is %lf \n", dbl);
   printf("Octal value is %o \n", no);
   printf("Hexadecimal value is %x \n", no);
   return 0;
```

SCANF() FUNCTION IN C LANGUAGE:

In C programming language, scanf() function is used to read character, string, numeric data from keyboard

Consider below example program where user enters a character. This value is assigned to the variable “ch” and then displayed.

Then, user enters a string and this value is assigned to the variable “str” and then displayed.



```
#include <stdio.h>
int main()
{
   char ch;
   char str[100];
   printf("Enter any character \n");
   scanf("%c", &ch);
   printf("Entered character is %c \n", ch);
   printf("Enter any string ( upto 100 character ) \n
   scanf("%s", &str);
   printf("Entered string is %s \n", str);
   return 0;
}
```

17. What is goto statement? Why it is not preferred in programs along with an example?

The goto statement is rarely used because it makes program confusing, less readable and complex. Also, when this is used, the control of the program won't be easy to trace, hence it makes testing and debugging difficult.

C – goto statement

When a goto statement is encountered in a C program, the control jumps directly to the label mentioned in the goto stateemnt

Syntax of goto statement in C

goto label_name;

..

..

label_name: C-statements

18. \ **Write the syntax of the following:**

i)if ii)if else iii)if..else..if iv)while v)dowhile vi)for

if statement:

if (condition)

{

statement-block;

}

if-else statement:

if (test expression) {

// statements to be executed if the test expression is true

}

else {

// statements to be executed if the test expression is false

}

if..else..if statement

if (test expression1) {

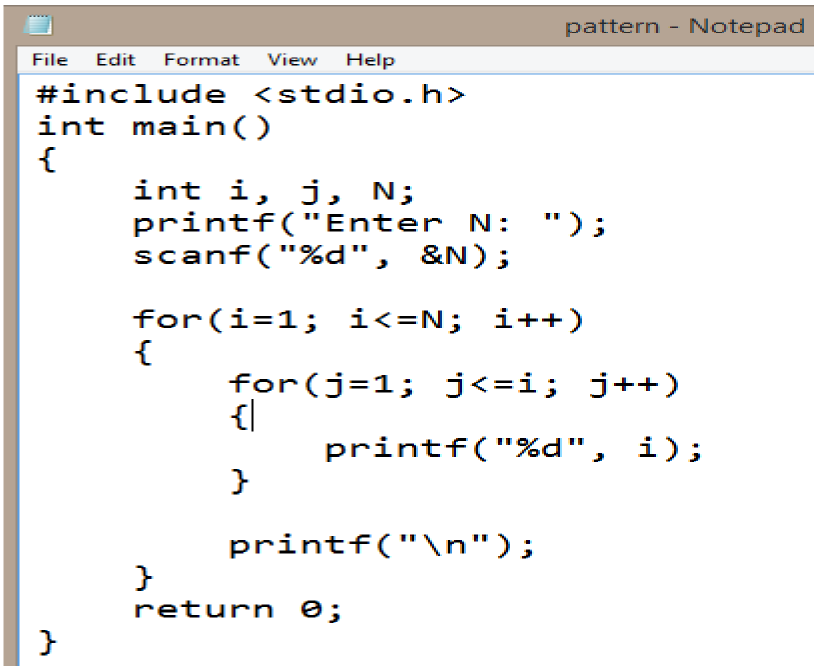
```
  // statement(s)
}
else if(test expression2) {
  // statement(s)
}
else if (test expression3) {
  // statement(s)
}
.
.
else {
  // statement(s)
}
```
while loop:
```
        while (testExpression)
{
   // statements inside the body of the loop
}
```

do-while loop:
```
        do
{
  // statements inside the body of the loop
}
while (testExpression);
```
for loop:
```
for (initializationStatement; testExpression; updateStatement)
{
   // statements inside the body of loop
}
```

19. Write a c program to print the following :

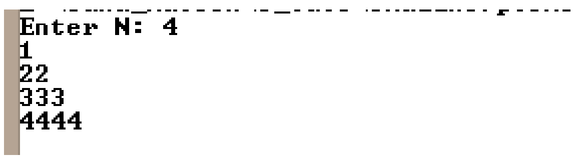
```
1
2  2
3  3  3
4  4  4  4
```

Program:



```c
#include <stdio.h>
int main()
{
	int i, j, N;
	printf("Enter N:  ");
	scanf("%d", &N);

	for(i=1; i<=N; i++)
	{
		for(j=1; j<=i; j++)
		{
			printf("%d", i);
		}

		printf("\n");
	}
	return 0;
}
```

Output:



```
Enter N: 4
1
22
333
4444
```

UNIT-2

Short Answer Questions for 2 Marks

1. **What is an Array?How do we declare an array?Mention the different types of array along with example?**

   An array is a group (or collection) of same data types. For example an int array holds the elements of int types while a float array holds the elements of float types.

   declare Array in C

   int num[35]; /* An integer array of 35 elements */

   char ch[10]; /* An array of characters for 10 elements */

   In c programming language, arrays are classified into two types. They are as follows...

   Single Dimensional Array / One Dimensional Array

   Multi Dimensional Array

2. **Write about two dimensional array with example?**

   An array of arrays is called as multi dimensional array. In simple words, an array created with more than one dimension (size) is called as multi dimensional array. Multi dimensional array can be of two dimensional array or three dimensional array or four dimensional array or more...
   Most popular and commonly used multi dimensional array is two dimensional array. The 2-D arrays are used to store data in the form of table. We also use 2-D arrays to create mathematical matrices.
   Declaration of Two Dimensional Array
   We use the following general syntax for declaring a two dimensional array...
   datatype arrayName [rowSize] [columnSize] ;
   Example Code
   int matrix_A [2][3] ;
   The above declaration of two dimensional array reserves 6 continuous memory locations of 2 bytes each in the form of 2 rows and 3 columns.

3. **Define string?What are the different ways to accept a string?**

In C programming, a string is a sequence of characters terminated with a null character \0. For example:

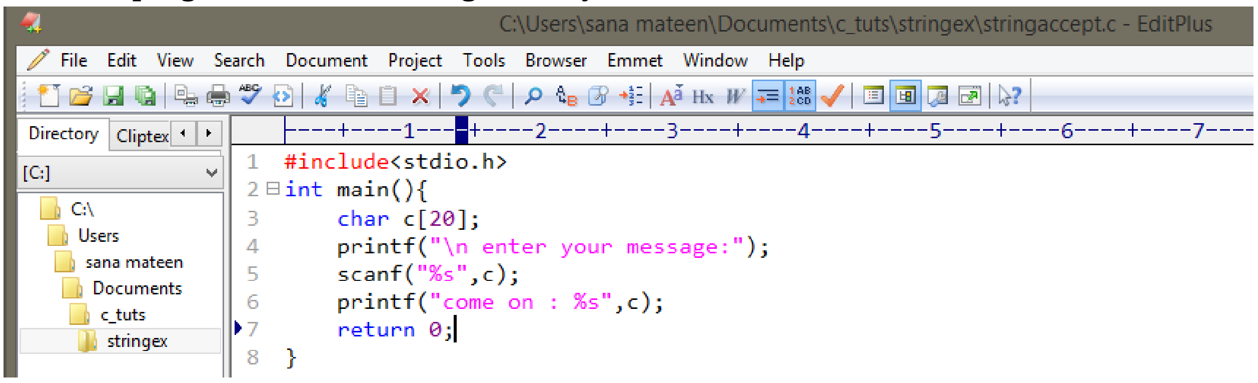
char c[] = “cse first";

different ways to accept a string are:

- use the scanf() function to read a string.
- The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab etc.)
- use the fgets() function to read a line of string. And, you can use puts() to display the string.
- **Syntax:**
- fgets(array_name, sizeof(array_name), stdin);

  puts(array_name);

4. Write a c program to handle string as array of characters?


C:\Users\sana mateen\Documents\c_tuts\stringex\stringaccept.c - EditPlus
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptex
[C:]
C:\
Users
sana mateen
Documents
c_tuts
stringex

```
#include<stdio.h>
int main(){
    char c[20];
    printf("\n enter your message:");
    scanf("%s",c);
    printf("come on : %s",c);
    return 0;
}
```

5. Write the syntax of the following
i)strlen ii)strcat iii)strcpy iv)strstr
i)strlen
size_t strlen(const char *str)
size_t represents unsigned short
It returns the length of the string without including end character (terminating char ‘\0’).

ii) strcat
char *strcat(char *str1, char *str2)
It concatenates two strings and returns the concatenated string.
iii)strcpy
char *strcpy(char *str1, char *str2)
It copies the string str2 into string str1, including the end character (terminator char '\0').
iv)strstr
char *strstr(char *str, char *srch_term)
It is similar to strchr, except that it searches for string srch_term instead of a single char.

Short Answer Questions for 3 Marks:

6. **Define structure and mention the syntax of it along with an example?**

It is a collection of related variables (aggregates) under one name.It can contain variables of different data types

Commonly used to define records to be stored in files

Combined with pointers, can create linked lists, stacks, queues, and trees

Syntax:

```
struct <structure name>

{

structure element 1 ;

structure element 2 ;

structure element 3 ;

......

......

} s1;
```

Example

```
struct Student {

    char name [20];

    char address [20];

    int rollno;
    char class[5];

    char year[20];
  };
```

7. Define union and mention the syntax of it along with an example?

A union is a special data type available in C that allows to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple-purpose.

Syntax:

```
union [union tag] {

  member definition;

  member definition;

  ...

  member definition;

} [one or more union variables];
```

Example:

```
union Data {

  int i;

  float f;

  char str[20];

} data;
```

8. Differentiate between structures and unions?

Structures	Unions
Struct keyword is used to declare the structure	Union keyword is used to declare the Union
Structure variable will allocate memory for all the structure members separately.	Union variable will allocate common memory for all the union members.
It allows us to access any or all the members at any time.	It allows us to access only one union member at a time.

9. **Define pointers and mention their advantages?**

A **pointer** is a variable that stores the address of another variable.The following are the advantages of pointers:

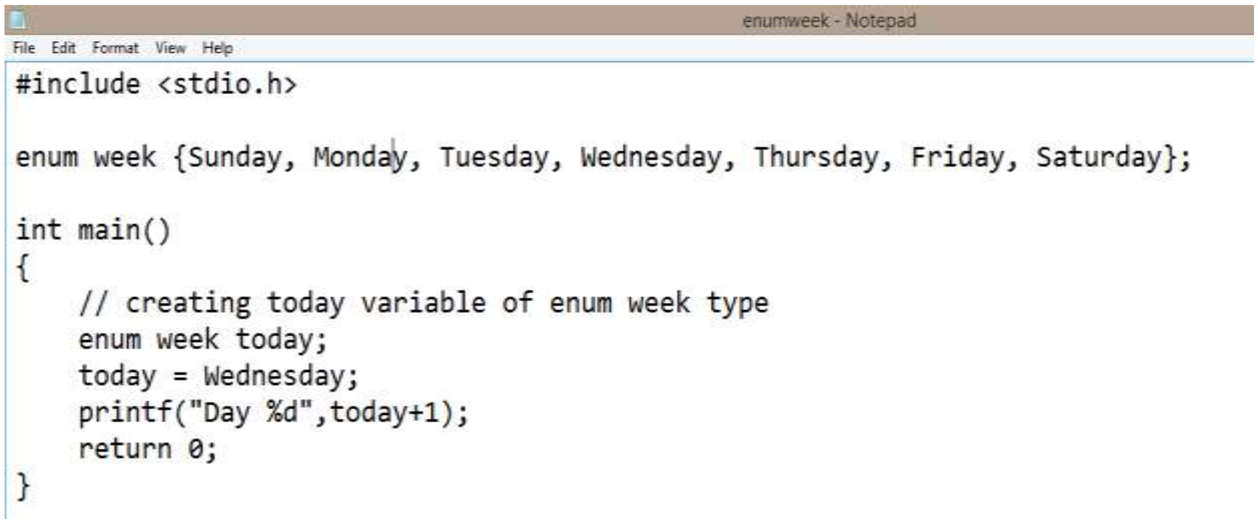
(i) It allows management of structures which are allocated memory dynamically.

(ii) It allows passing of arrays and strings to functions more efficiently.

(iii) It makes possible to pass address of structure instead of entire structure to the functions.

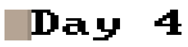
10. Write a c program to implement enumerated data type?

Program



```
#include <stdio.h>

enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

int main()
{
    // creating today variable of enum week type
    enum week today;
    today = Wednesday;
    printf("Day %d",today+1);
    return 0;
}
```

Output:



```
Day 4
```

11. List the usage of self referential structures in Linked List?

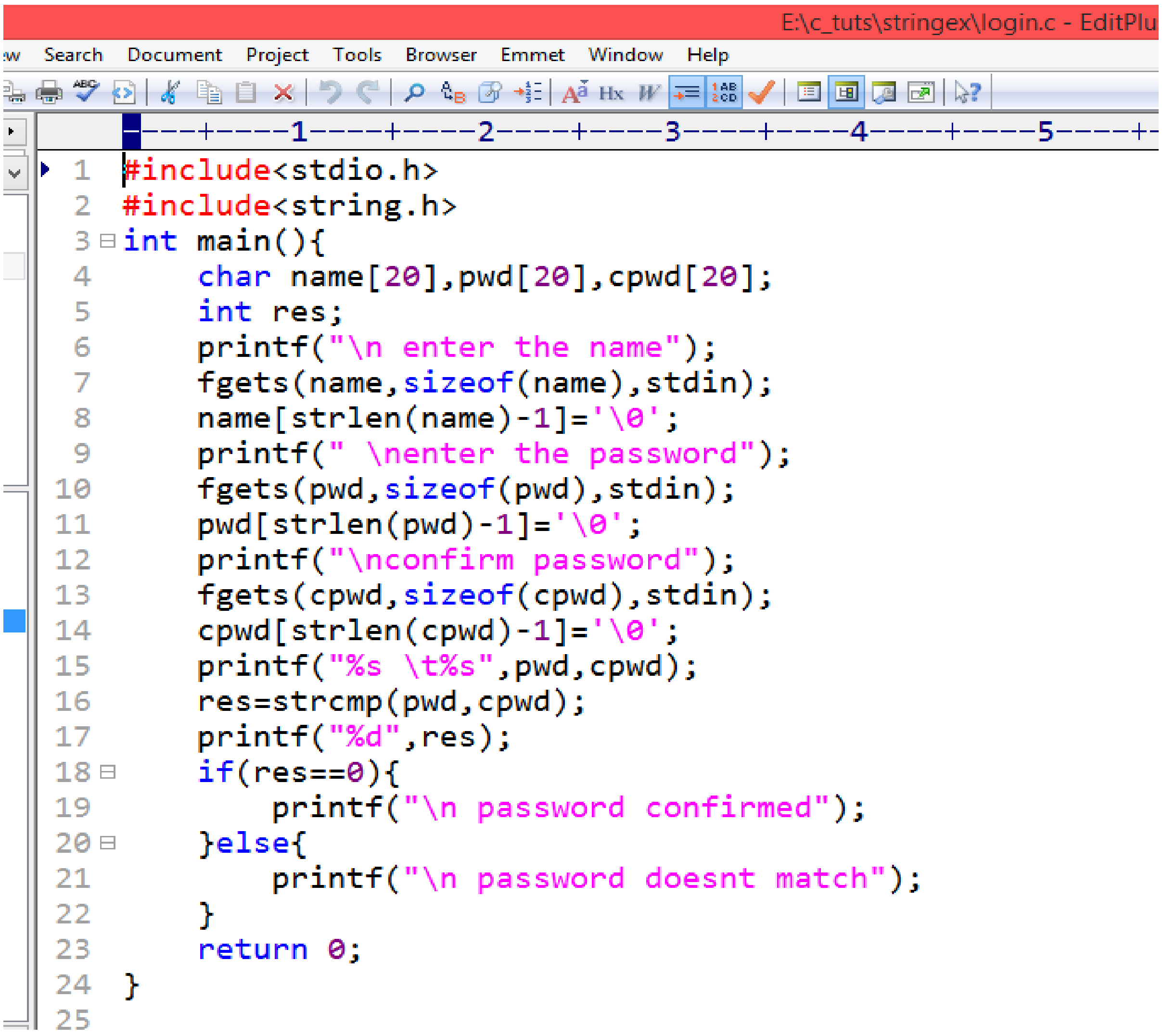
A self-referential structure is one of the data structures which refer to the pointer to (points) to another structure of the same type. For example, a linked list is supposed to be a self-referential data structure. The next node of a node is being pointed, which is of the same struct type. For example,

```
typedef struct listnode {
void *data;
struct listnode *next;
} linked_list;
```

In the above example, the listnode is a self-referential structure – because the *next is of the type struct listnode.

12. Write a c program to compare two strings?

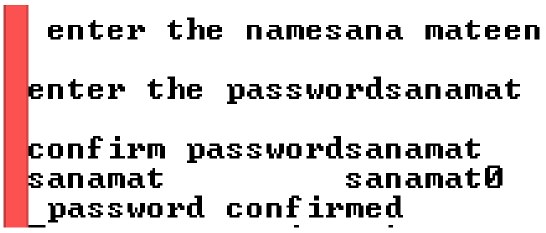
Program:



E:\c_tuts\stringex\login.c - EditPlu

```c
#include<stdio.h>
#include<string.h>
int main(){
	char name[20],pwd[20],cpwd[20];
	int res;
	printf("\n enter the name");
	fgets(name,sizeof(name),stdin);
	name[strlen(name)-1]='\0';
	printf(" \nenter the password");
	fgets(pwd,sizeof(pwd),stdin);
	pwd[strlen(pwd)-1]='\0';
	printf("\nconfirm password");
	fgets(cpwd,sizeof(cpwd),stdin);
	cpwd[strlen(cpwd)-1]='\0';
	printf("%s \t%s",pwd,cpwd);
	res=strcmp(pwd,cpwd);
	printf("%d",res);
	if(res==0){
		printf("\n password confirmed");
	}else{
		printf("\n password doesnt match");
	}
	return 0;
}
```

Output:



```
 enter the namesana mateen

enter the passwordsanamat

confirm passwordsanamat
sanamat         sanamat0
 password confirmed
```

Long Answer Questions for 5 Marks:

1. **What is an Array?List the different types of array?Write a c program to display the elements of the array?**

An array is a collection of data that holds fixed number of values of same type.

For example: if you want to store marks of 100 students, you can create an array for it.

float marks[100];

The length and type of an array cannot be changed after array declaration.

Here, we declared an array, mark, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

How to declare an array in C?

data_type array_name[array_size];

For example,

float mark[5];

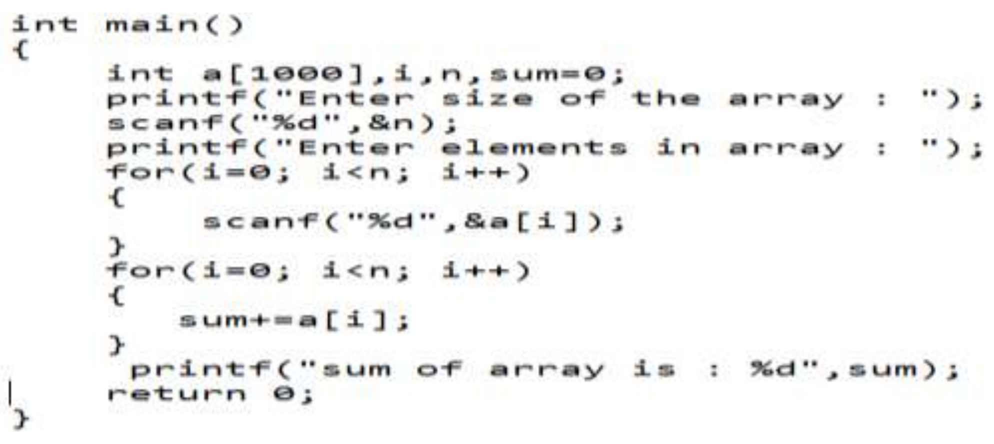
In C, arrays can be classified based on how the data items are arranged for human understanding. Arrays are broadly classified into three categories,

1. One Dimensional Arrays

2. Two Dimensional Arrays

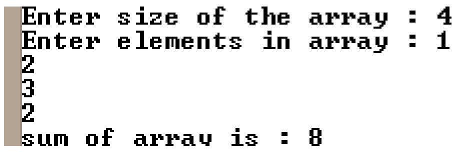
3. Multi Dimensional Arrays

Program:



```
int main()
{
    int a[1000],i,n,sum=0;
    printf("Enter size of the array : ");
    scanf("%d",&n);
    printf("Enter elements in array : ");
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0; i<n; i++)
    {
       sum+=a[i];
    }
     printf("sum of array is : %d",sum);
    return 0;
}
```

ut:



```
Enter size of the array : 4
Enter elements in array : 1
2
3
2
sum of array is : 8
```

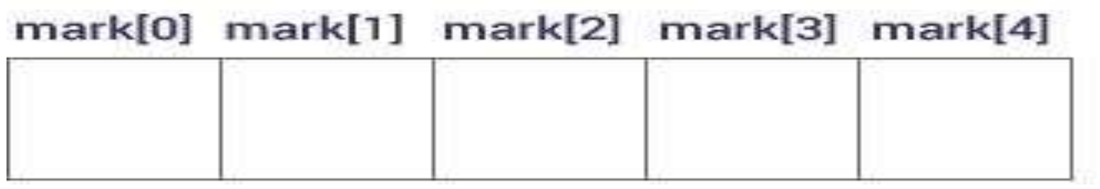
2. Explain in detail creation and manipulation of elements in an array?

You can access elements of an array by an index.

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

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 1000. Then, the next address, a[1], will be 1004, address of a[2] will be 1008 and so on. It's because the size of a float is 4 bytes.

Creation of an consists two things: Element Type and Array Size.

Element Type: What kind of data an array can hold?

An array can hold any one of the following data:

integer, double, character data.

Array Size: How many elements an array can contain?

Once an array size is defined it cannot be changed at run-time.

Array Manipulation

The most convenient way of performing array manipulation is to use the for repetitive construct (for loop) for accessing elements of the array.

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

```
double salary = balance[9];
```

The above statement will take the 10th element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays −

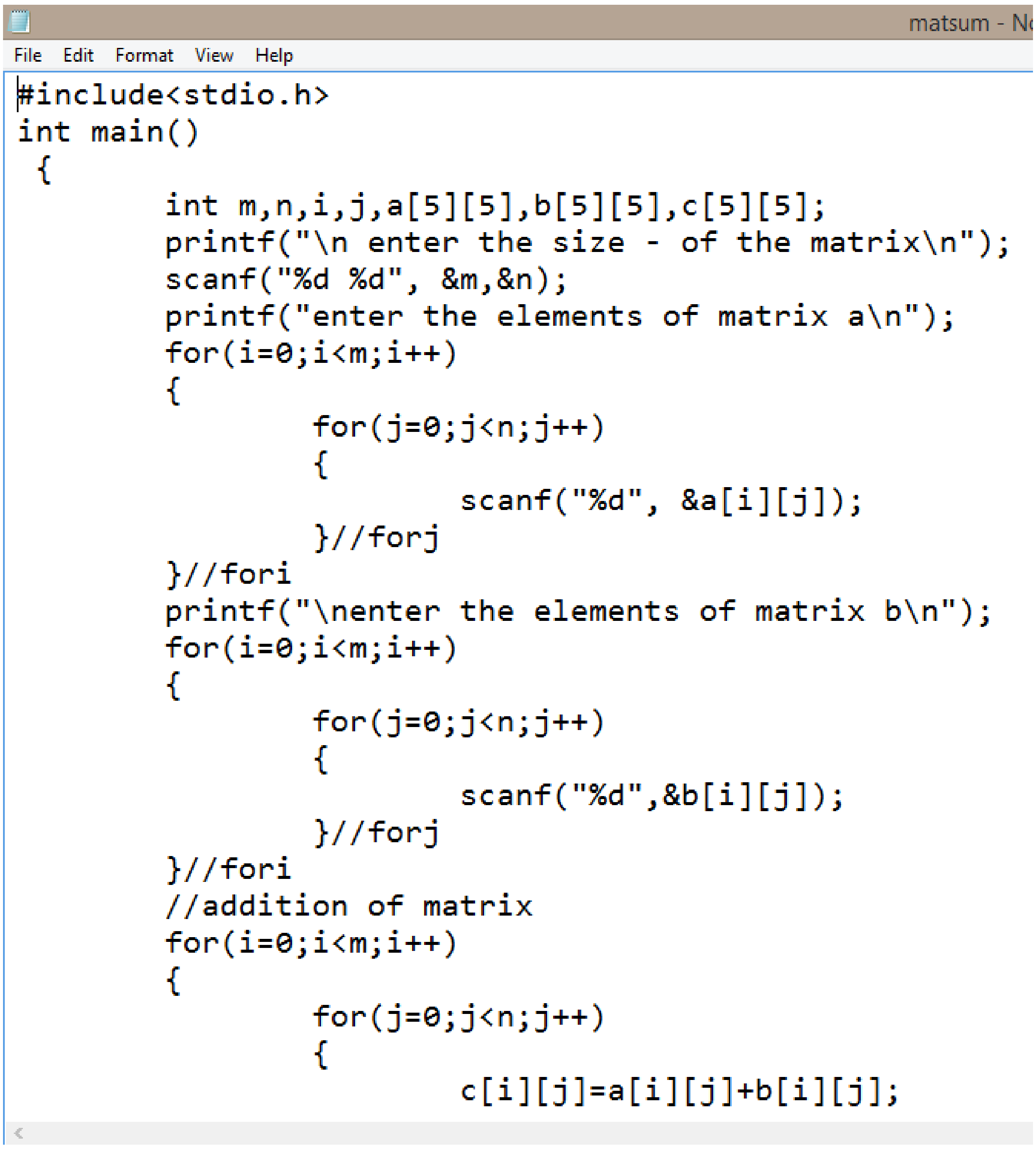
```
#include <stdio.h>

int main () {

   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ ) {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

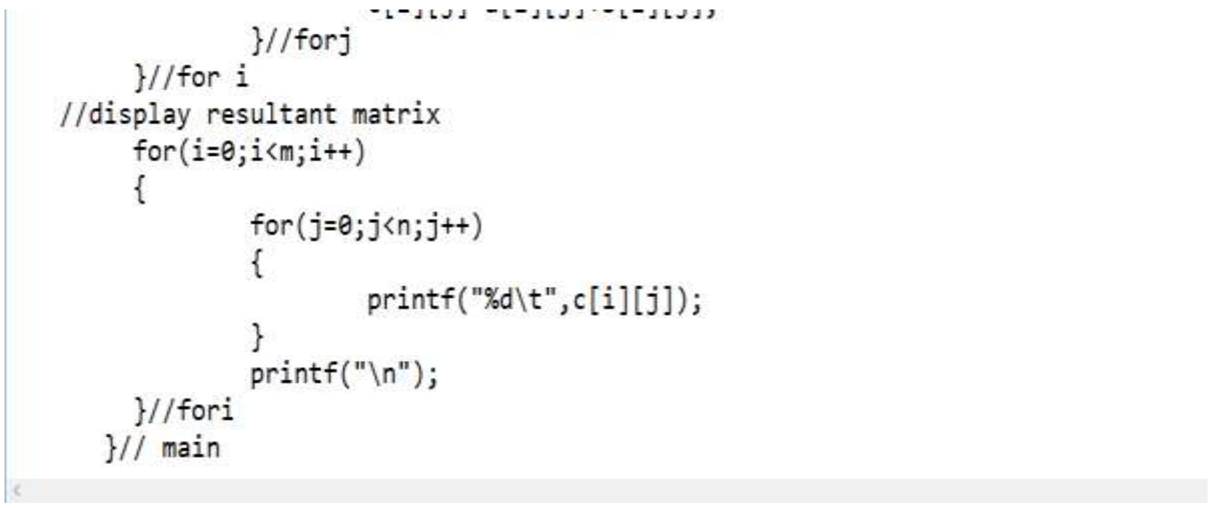
```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

3. Write a c program to calculate addition of two matrices?

Program:

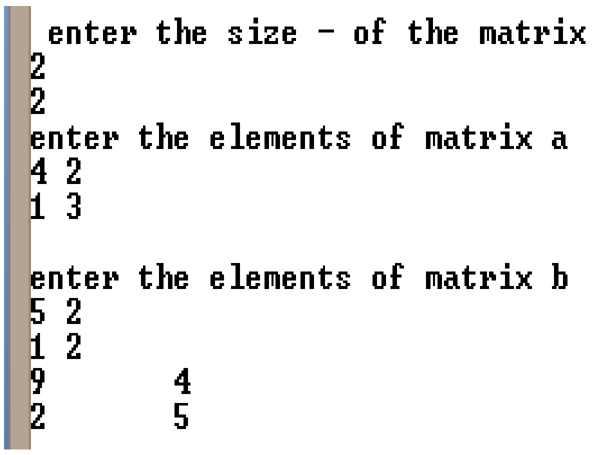


```
#include<stdio.h>
int main()
 {
        int m,n,i,j,a[5][5],b[5][5],c[5][5];
        printf("\n enter the size - of the matrix\n");
        scanf("%d %d", &m,&n);
        printf("enter the elements of matrix a\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d", &a[i][j]);
                }//forj
        }//fori
        printf("\nenter the elements of matrix b\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",&b[i][j]);
                }//forj
        }//fori
        //addition of matrix
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        c[i][j]=a[i][j]+b[i][j];
```



```
            }//forj
      }//for i
  //display resultant matrix
      for(i=0;i<m;i++)
      {
            for(j=0;j<n;j++)
            {
                  printf("%d\t",c[i][j]);
            }
            printf("\n");
      }//fori
    }// main
```

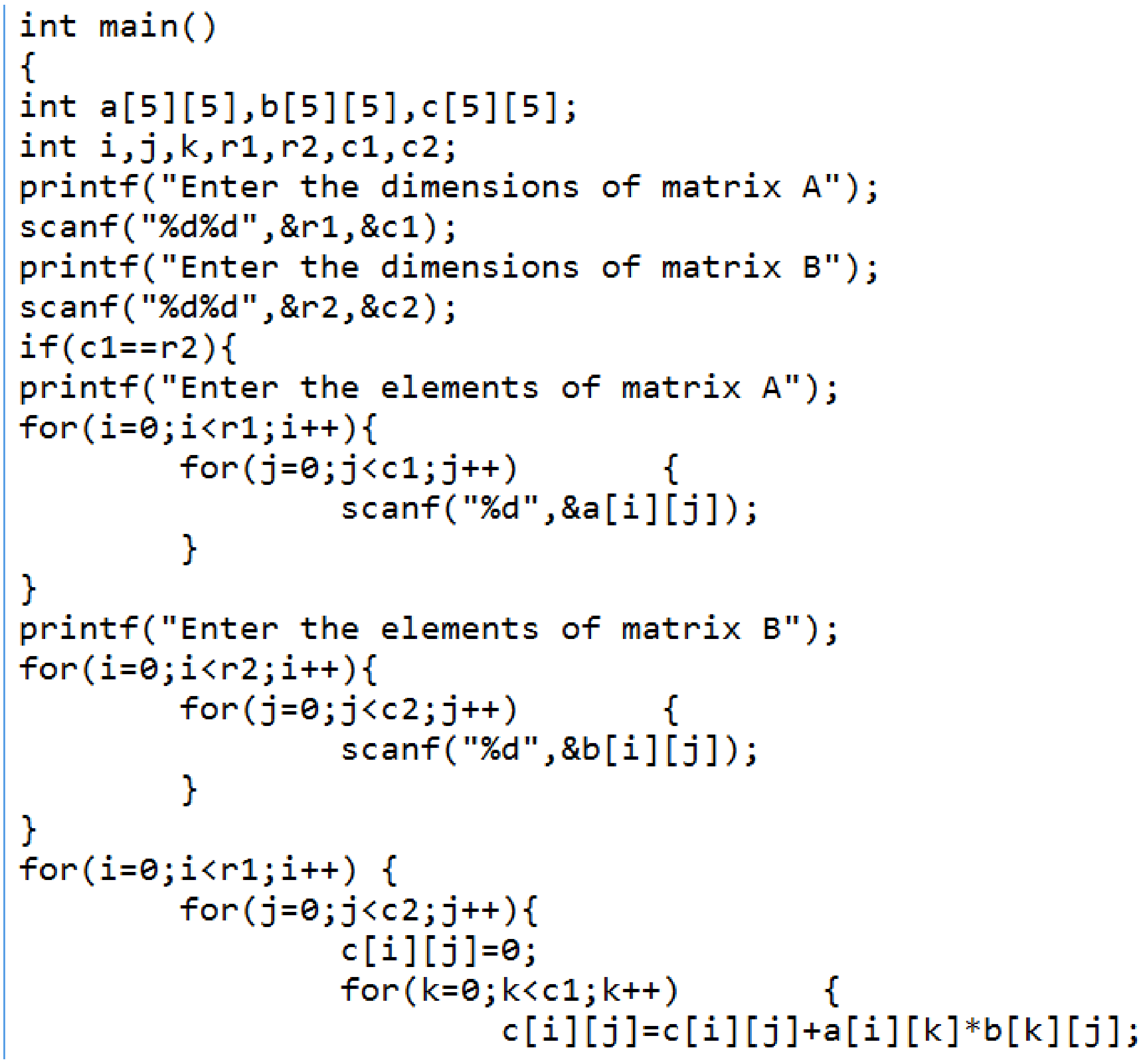
Output:



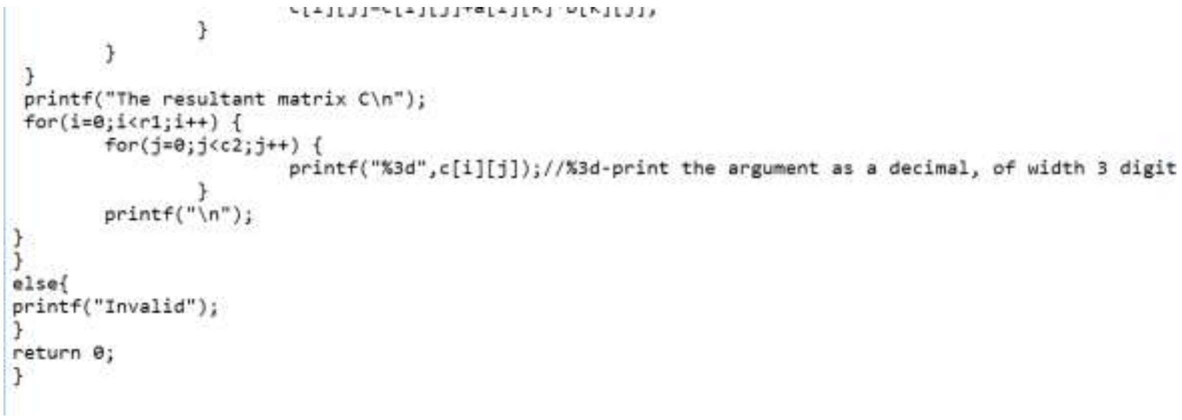
```
 enter the size - of the matrix
2
2
enter the elements of matrix a
4 2
1 3

enter the elements of matrix b
5 2
1 2
9       4
2       5
```

4. Write a c program to multiply two matrices?

Program:

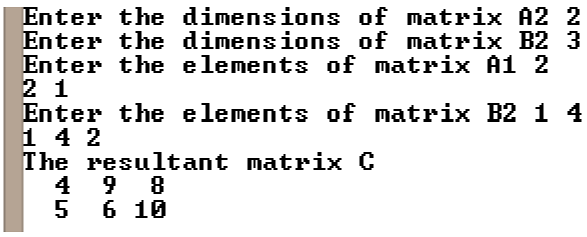


```
int main()
{
int a[5][5],b[5][5],c[5][5];
int i,j,k,r1,r2,c1,c2;
printf("Enter the dimensions of matrix A");
scanf("%d%d",&r1,&c1);
printf("Enter the dimensions of matrix B");
scanf("%d%d",&r2,&c2);
if(c1==r2){
printf("Enter the elements of matrix A");
for(i=0;i<r1;i++){
        for(j=0;j<c1;j++)       {
                scanf("%d",&a[i][j]);
        }
}
printf("Enter the elements of matrix B");
for(i=0;i<r2;i++){
        for(j=0;j<c2;j++)       {
                scanf("%d",&b[i][j]);
        }
}
for(i=0;i<r1;i++) {
        for(j=0;j<c2;j++){
                c[i][j]=0;
                for(k=0;k<c1;k++)       {
                        c[i][j]=c[i][j]+a[i][k]*b[k][j];
```



```
                }
        }
}
printf("The resultant matrix C\n");
for(i=0;i<r1;i++) {
        for(j=0;j<c2;j++) {
                printf("%3d",c[i][j]);//%3d-print the argument as a decimal, of width 3 digit
                }
        printf("\n");
}
}
else{
printf("Invalid");
}
return 0;
}
```

Output:



```
Enter the dimensions of matrix A2 2
Enter the dimensions of matrix B2 3
Enter the elements of matrix A1 2
2 1
Enter the elements of matrix B2 1 4
1 4 2
The resultant matrix C
  4  9  8
  5  6 10
```

5. **Define Strings?Explain initialization of String and accepting the string along with an example program?**

In C programming, a string is a sequence of characters terminated with a null character \0. For example:

char c[] = “cse first";

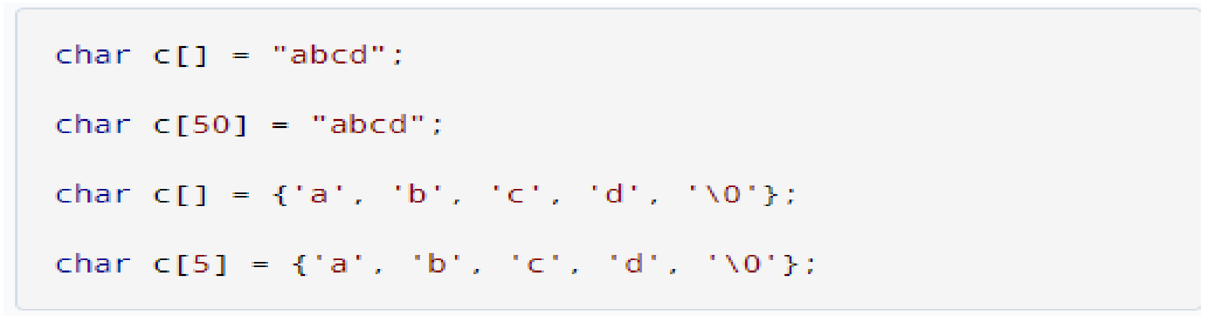
c	s	e		f	i	r	s	t	\0

Declare String

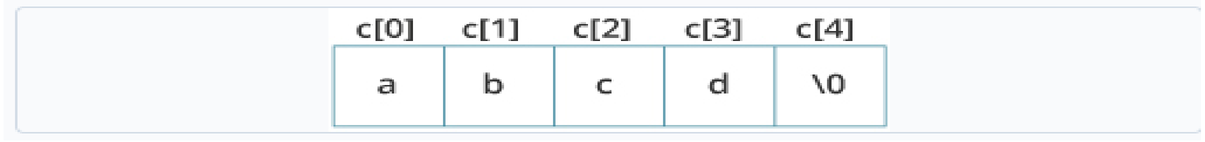
char s[5];

S[0]	**S[1]**	**S[2]**	**S[3]**	**S[4]**
s	**a**	**n**	**a**	**\0**

Initialize strings



```
char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



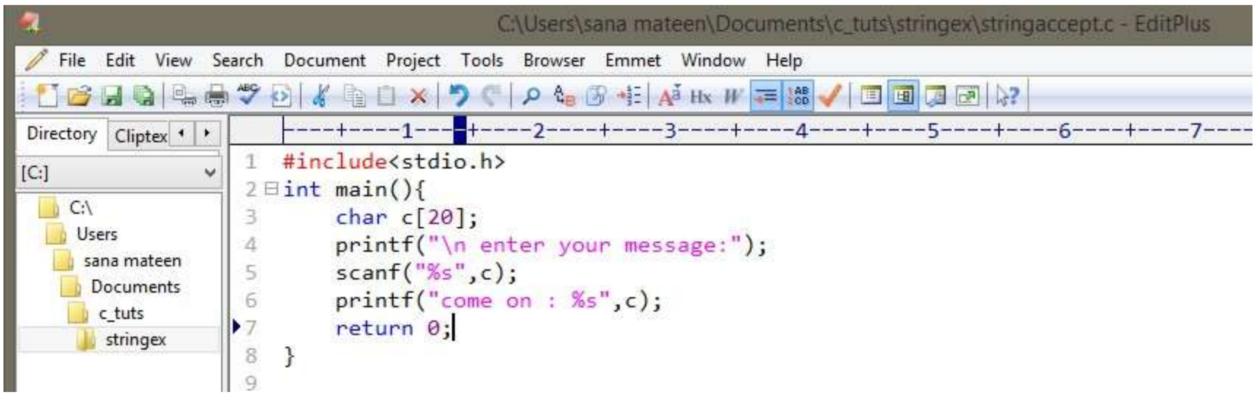
c[0]	c[1]	c[2]	c[3]	c[4]
a	b	c	d	\0

char c[11]=“sana mateen”;

Here, we are trying to assign 12 characters (the last character is '\0') to a char array having 11 characters. This is bad and you should never do this.

- You can use the scanf() function to read a string.
- The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab etc.)

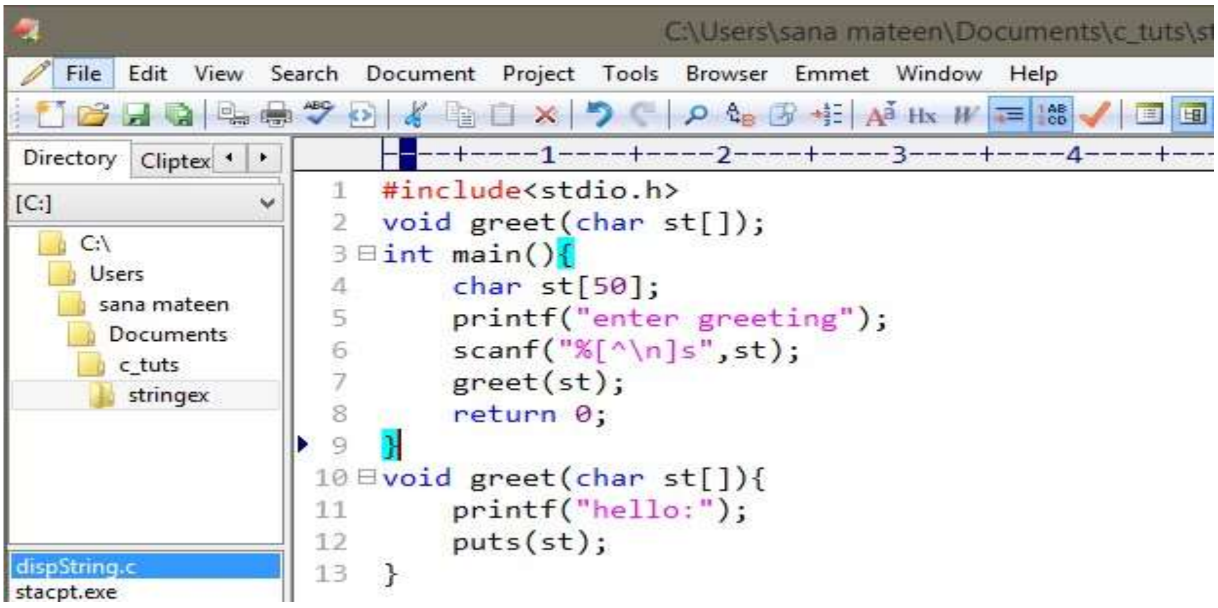
Program:


C:\Users\sana mateen\Documents\c_tuts\stringex\stringaccept.c - EditPlus
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptex
[C:]
C:\
Users
sana mateen
Documents
c_tuts
stringex

```
#include<stdio.h>
int main(){
    char c[20];
    printf("\n enter your message:");
    scanf("%s",c);
    printf("come on : %s",c);
    return 0;
}
```

6. **Write a c program to pass strings to functions?**

Program:



C:\Users\sana mateen\Documents\c_tuts\s
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptex
[C:]
C:\
Users
sana mateen
Documents
c_tuts
stringex
dispString.c
stacpt.exe

```
#include<stdio.h>
void greet(char st[]);
int main(){
    char st[50];
    printf("enter greeting");
    scanf("%[^\n]s",st);
    greet(st);
    return 0;
}
void greet(char st[]){
    printf("hello:");
    puts(st);
}
```

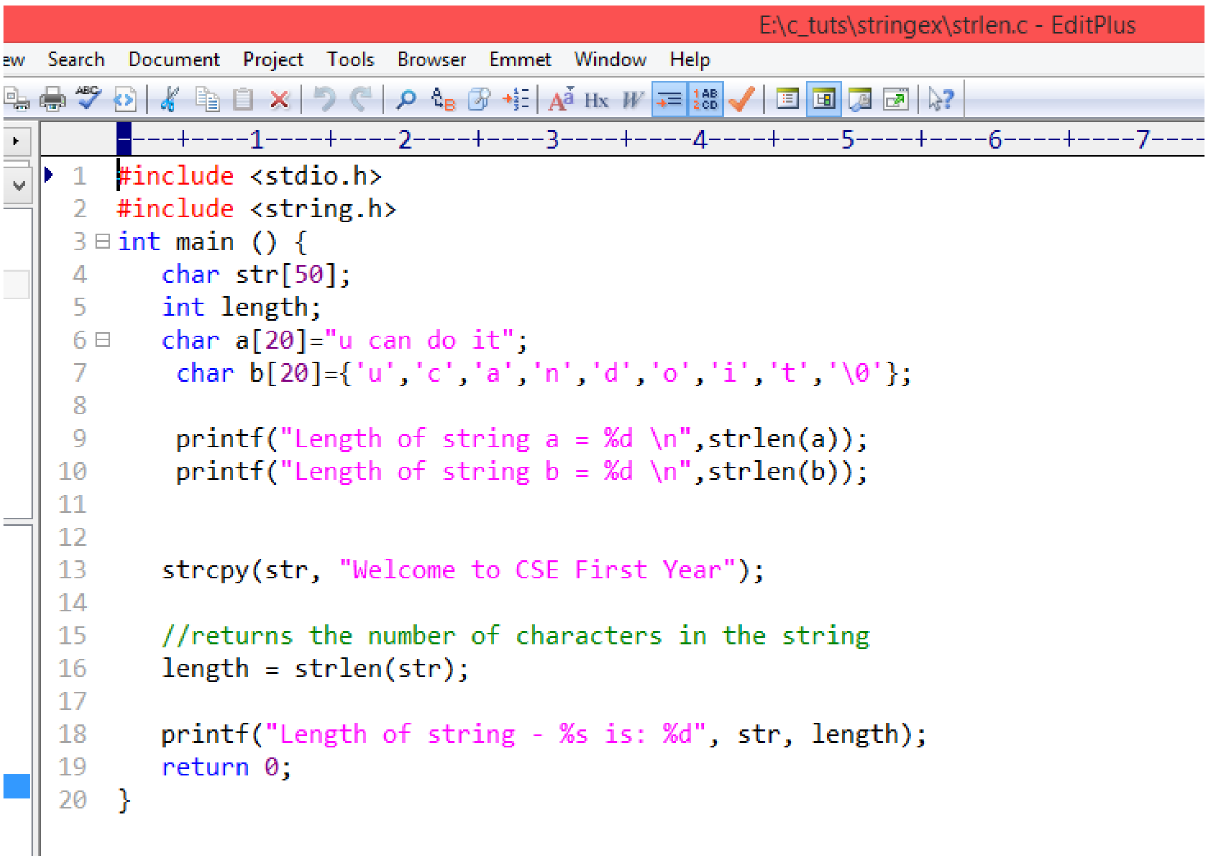
Output:

```
enter greetinghello there
hello:hello there
```

7. **Write about the following:**

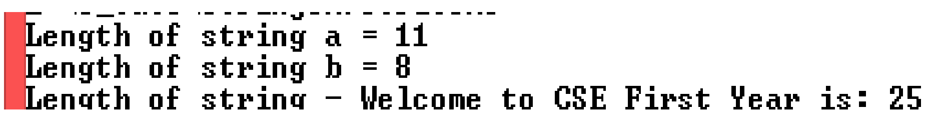
i)strlen ii)strcat iii)strcpy iv)strstr

i)strlen
size_t strlen(const char *str)
size_t represents unsigned short
It returns the length of the string without including end character (terminating char '\0').
Program:


E:\c_tuts\stringex\strlen.c - EditPlus
ew Search Document Project Tools Browser Emmet Window Help

```c
#include <stdio.h>
#include <string.h>
int main () {
   char str[50];
   int length;
   char a[20]="u can do it";
    char b[20]={'u','c','a','n','d','o','i','t','\0'};

    printf("Length of string a = %d \n",strlen(a));
    printf("Length of string b = %d \n",strlen(b));


   strcpy(str, "Welcome to CSE First Year");

   //returns the number of characters in the string
   length = strlen(str);

   printf("Length of string - %s is: %d", str, length);
   return 0;
}
```

Output:



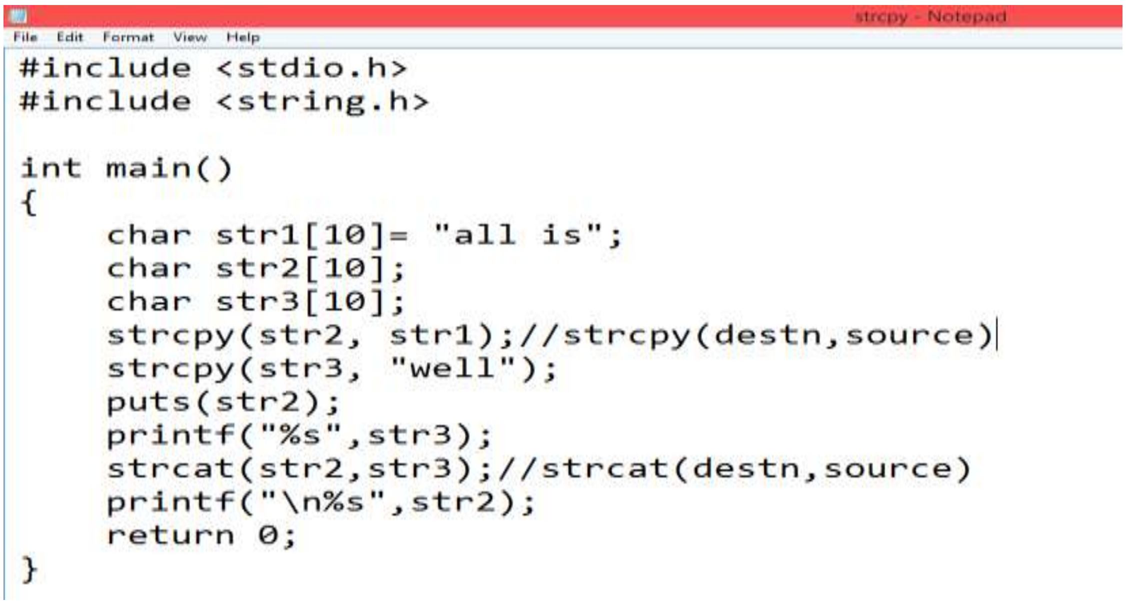
```
Length of string a = 11
Length of string b = 8
Length of string - Welcome to CSE First Year is: 25
```

ii) strcat

char *strcat(char *str1, char *str2)

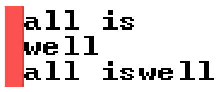
It concatenates two strings and returns the concatenated string.

Program:



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

int main()
{
	char str1[10]= "all is";
	char str2[10];
	char str3[10];
	strcpy(str2, str1);//strcpy(destn,source)
	strcpy(str3, "well");
	puts(str2);
	printf("%s",str3);
	strcat(str2,str3);//strcat(destn,source)
	printf("\n%s",str2);
	return 0;
}
```

Output:



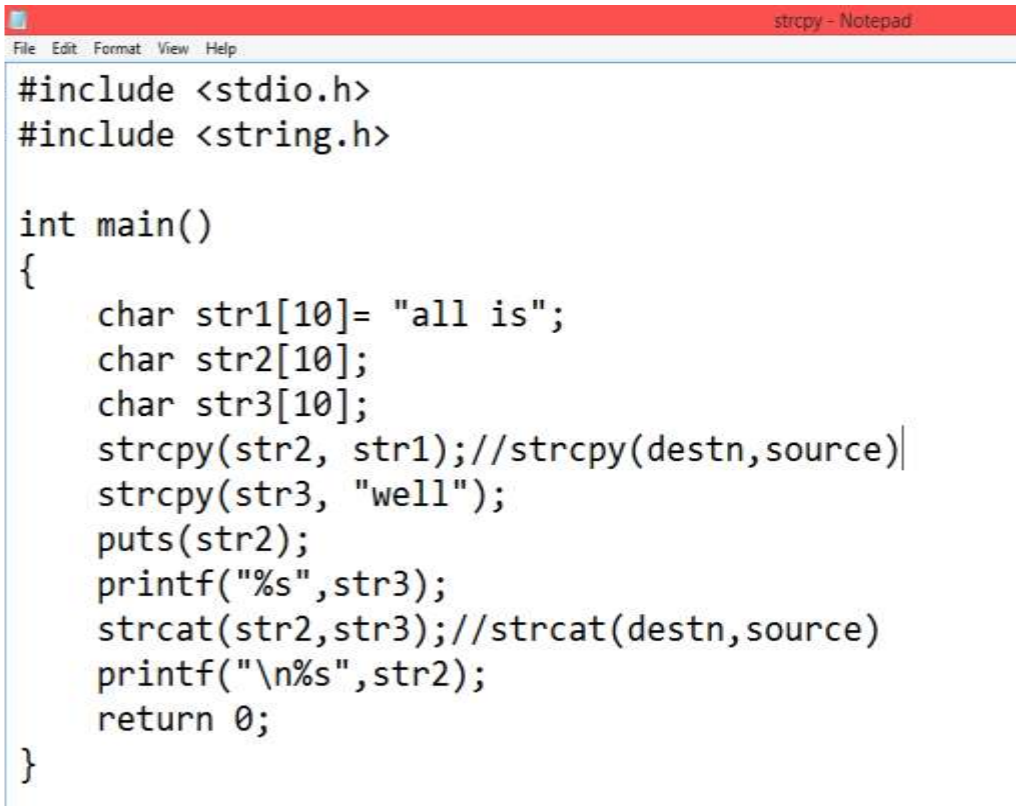
```
all is
well
all iswell
```

iii) strcpy

char *strcpy(char *str1, char *str2)

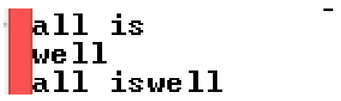
It copies the string str2 into string str1, including the end character (terminator char ‘\0’).

Program:



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

int main()
{
    char str1[10]= "all is";
    char str2[10];
    char str3[10];
    strcpy(str2, str1);//strcpy(destn,source)
    strcpy(str3, "well");
    puts(str2);
    printf("%s",str3);
    strcat(str2,str3);//strcat(destn,source)
    printf("\n%s",str2);
    return 0;
}
```

Output:



```
all is
well
all iswell
```

iv)strstr

char *strstr(char *str, char *srch_term)

It is similar to strchr, except that it searches for string srch_term instead of a single char.

Program:

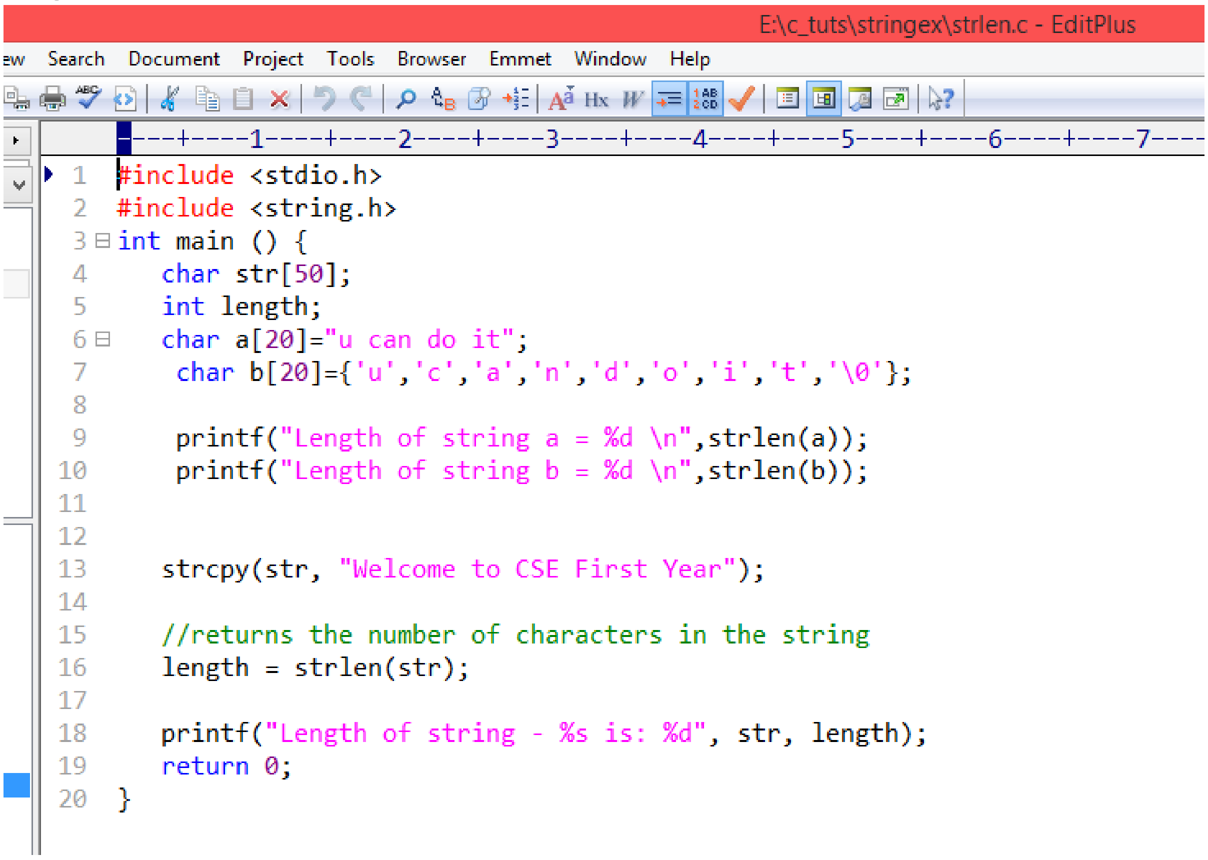
```c
#include <stdio.h>
#include <string.h>
int main()
{
        char inputstr[70] = "Lazy people get no where in life";
        printf ("Output string is: %s", strstr(inputstr, "Lazy"))
        return 0;
}
```

Output:

```
Output string is: Lazy people get no where in life
```

8. **Write a c program to implement strcpy and strlen functions?**

Program:


E:\c_tuts\stringex\strlen.c - EditPlus
ew Search Document Project Tools Browser Emmet Window Help

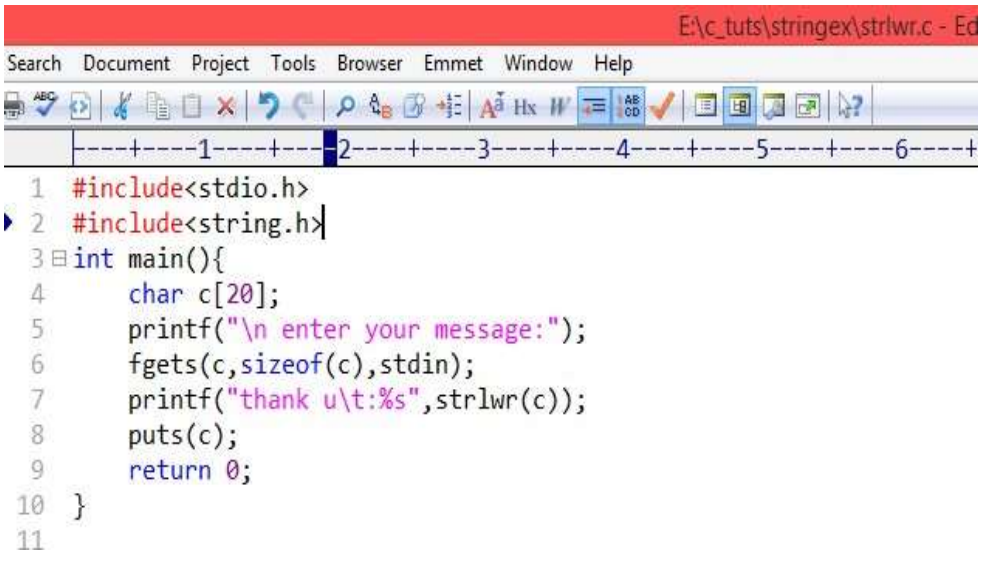
```c
#include <stdio.h>
#include <string.h>
int main () {
   char str[50];
   int length;
   char a[20]="u can do it";
    char b[20]={'u','c','a','n','d','o','i','t','\0'};

    printf("Length of string a = %d \n",strlen(a));
    printf("Length of string b = %d \n",strlen(b));


   strcpy(str, "Welcome to CSE First Year");

   //returns the number of characters in the string
   length = strlen(str);

   printf("Length of string - %s is: %d", str, length);
   return 0;
}
```

Output:

```
Length of string a = 11
Length of string b = 8
Length of string - Welcome to CSE First Year is: 25
```

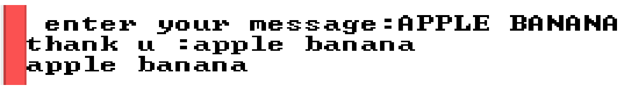
9. Write a c program to implement strlwr function?

Program:


E:\c_tuts\stringex\strlwr.c - Ed
Search Document Project Tools Browser Emmet Window Help

```c
#include<stdio.h>
#include<string.h>
int main(){
    char c[20];
    printf("\n enter your message:");
    fgets(c,sizeof(c),stdin);
    printf("thank u\t:%s",strlwr(c));
    puts(c);
    return 0;
}
```

Output:



```
 enter your message:APPLE BANANA
thank u :apple banana
apple banana
```

10. **What are structures?Write a c program to initialize and display the elements of structures?**

Collections of related variables (aggregates) under one name,It Can contain variables of different data types.

Commonly used to define records to be stored in files

Combined with pointers, can create linked lists, stacks, queues, and trees

Syntax:

```
struct <structure name>

{

structure element 1 ;

structure element 2 ;

structure element 3 ;

......

......

} ;
```

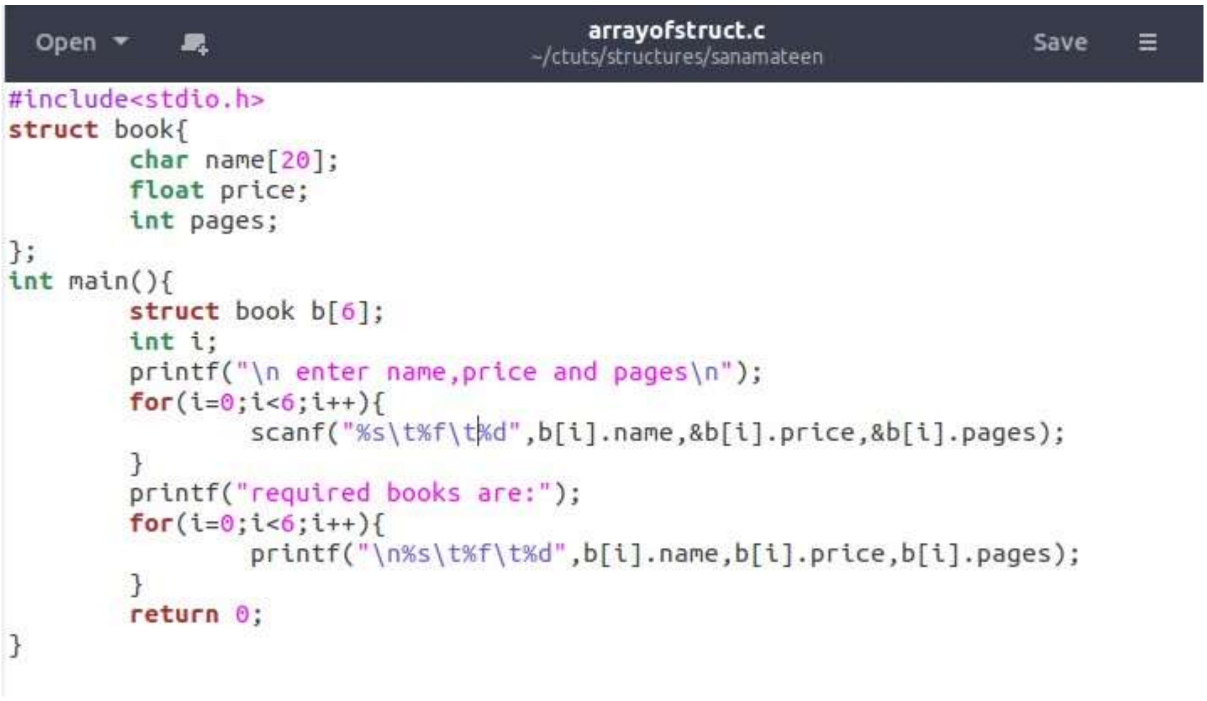
Example

```
            struct Student {

           char name [20];

           char address [20];

           int rollno;

           char class[5];

           char year[20];
       };
```

struct introduces the definition for structure Student

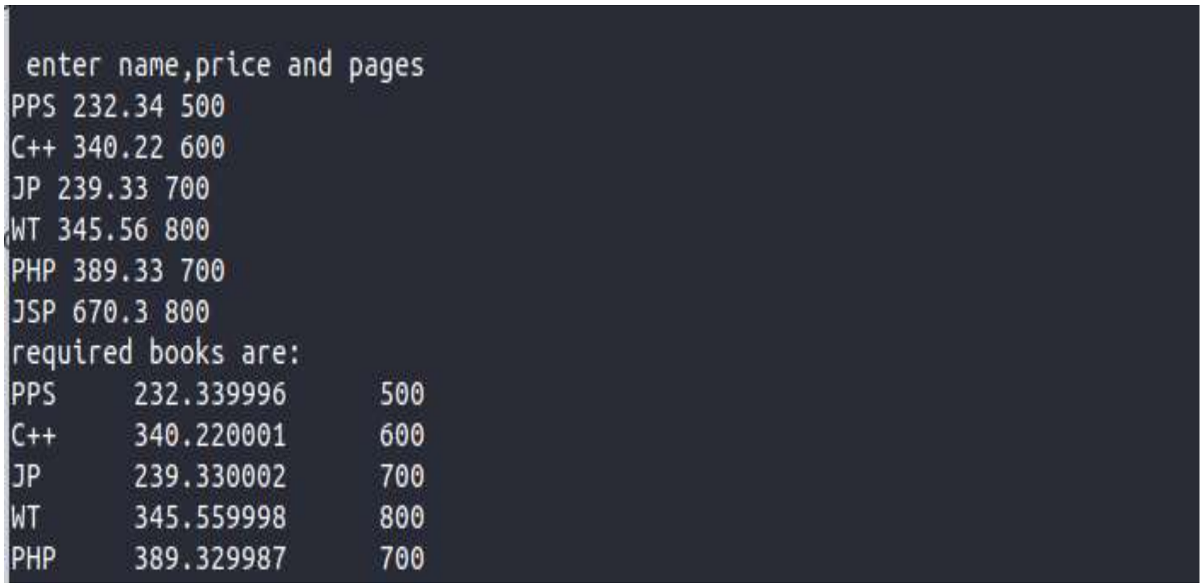
Student is the structure name and is used to declare variables of the structure type

Program:



```c
#include<stdio.h>
struct book{
	char name[20];
	float price;
	int pages;
};
int main(){
	struct book b[6];
	int i;
	printf("\n enter name,price and pages\n");
	for(i=0;i<6;i++){
		scanf("%s\t%f\t%d",b[i].name,&b[i].price,&b[i].pages);
	}
	printf("required books are:");
	for(i=0;i<6;i++){
		printf("\n%s\t%f\t%d",b[i].name,b[i].price,b[i].pages);
	}
	return 0;
}
```

Output:

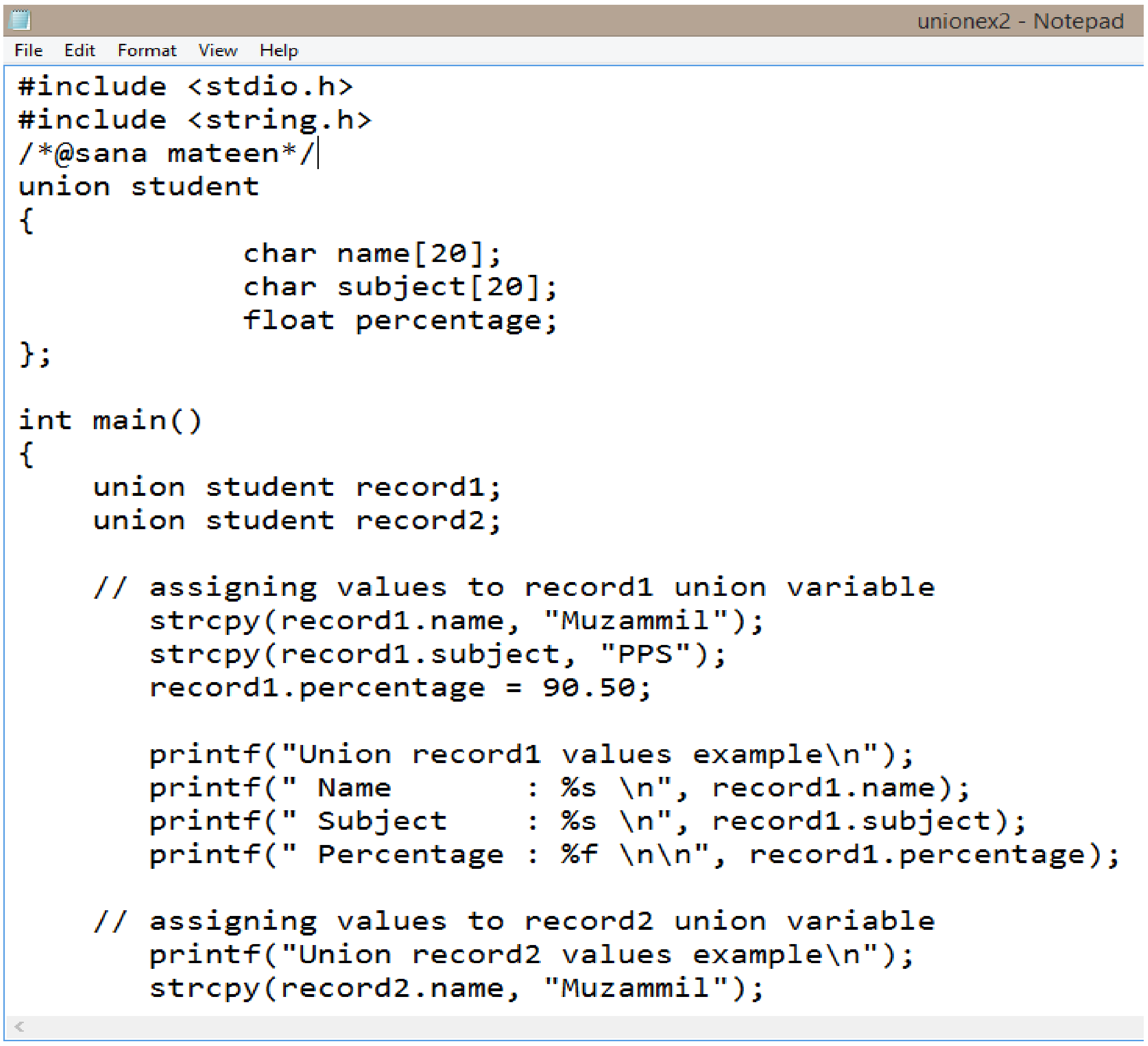


```
 enter name,price and pages
PPS 232.34 500
C++ 340.22 600
JP 239.33 700
WT 345.56 800
PHP 389.33 700
JSP 670.3 800
required books are:
PPS     232.339996      500
C++     340.220001      600
JP      239.330002      700
WT      345.559998      800
PHP     389.329987      700
```

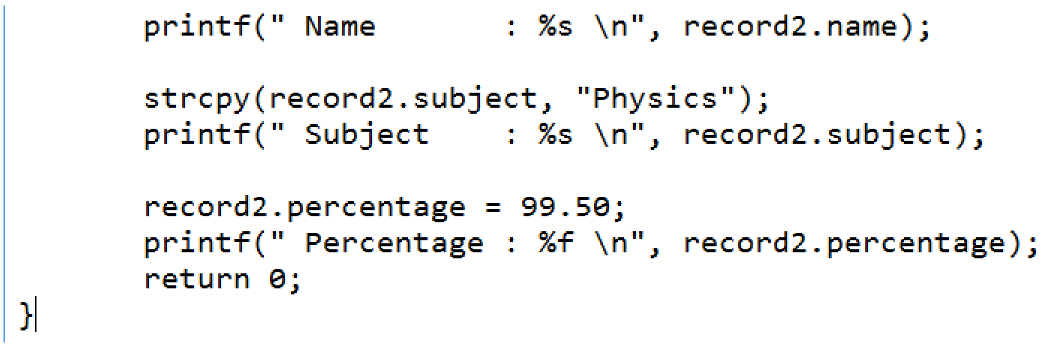
11. What are unions? Write a c program to initialize and display the elements of unions?

A union is a user-defined type similar to structs in C except for one key difference. Structs allocate enough space to store all its members wheres unions allocate the space to store only the largest member.

Program:



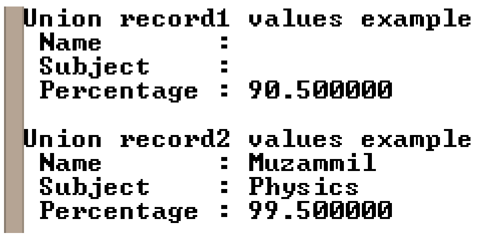
```
#include <stdio.h>
#include <string.h>
/*@sana mateen*/
union student
{
            char name[20];
            char subject[20];
            float percentage;
};

int main()
{
    union student record1;
    union student record2;

    // assigning values to record1 union variable
       strcpy(record1.name, "Muzammil");
       strcpy(record1.subject, "PPS");
       record1.percentage = 90.50;

       printf("Union record1 values example\n");
       printf(" Name       : %s \n", record1.name);
       printf(" Subject    : %s \n", record1.subject);
       printf(" Percentage : %f \n\n", record1.percentage);

    // assigning values to record2 union variable
       printf("Union record2 values example\n");
       strcpy(record2.name, "Muzammil");
```



```
       printf(" Name       : %s \n", record2.name);

       strcpy(record2.subject, "Physics");
       printf(" Subject    : %s \n", record2.subject);

       record2.percentage = 99.50;
       printf(" Percentage : %f \n", record2.percentage);
       return 0;
}
```

Output:

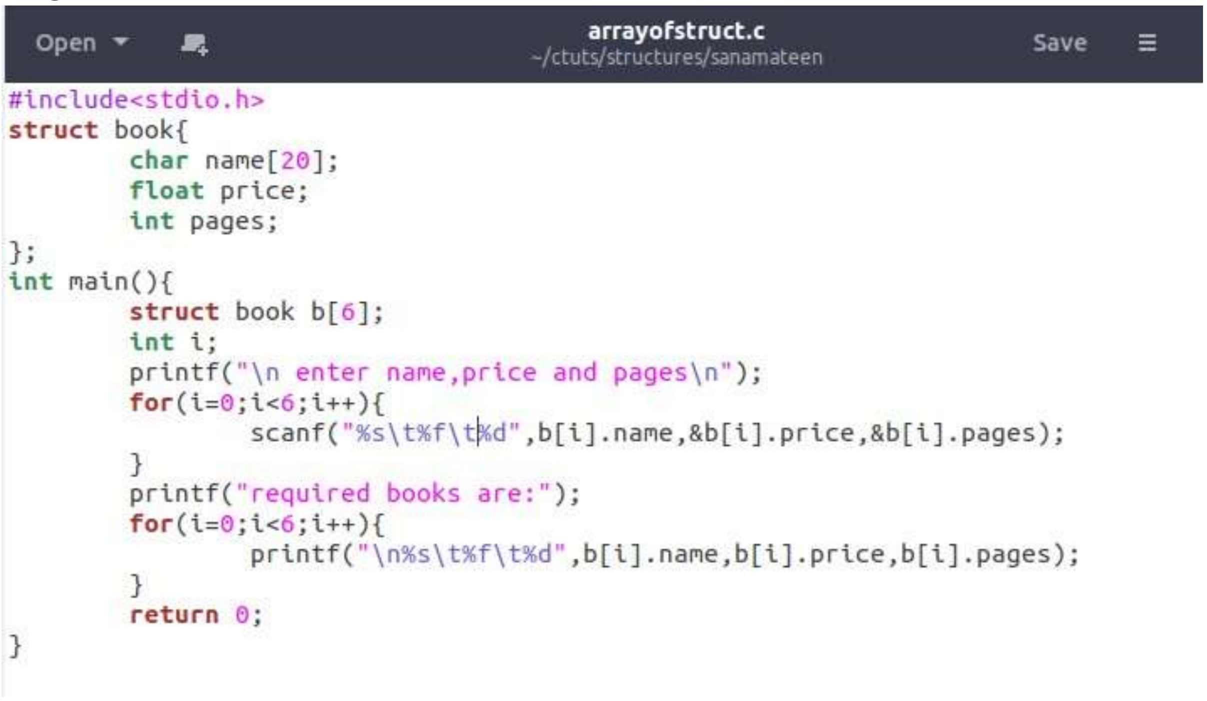
12. Explain Array of Structures in detail along with an example program?



```
Union record1 values example
 Name       :
 Subject    :
 Percentage : 90.500000

Union record2 values example
 Name       : Muzammil
 Subject    : Physics
 Percentage : 99.500000
```

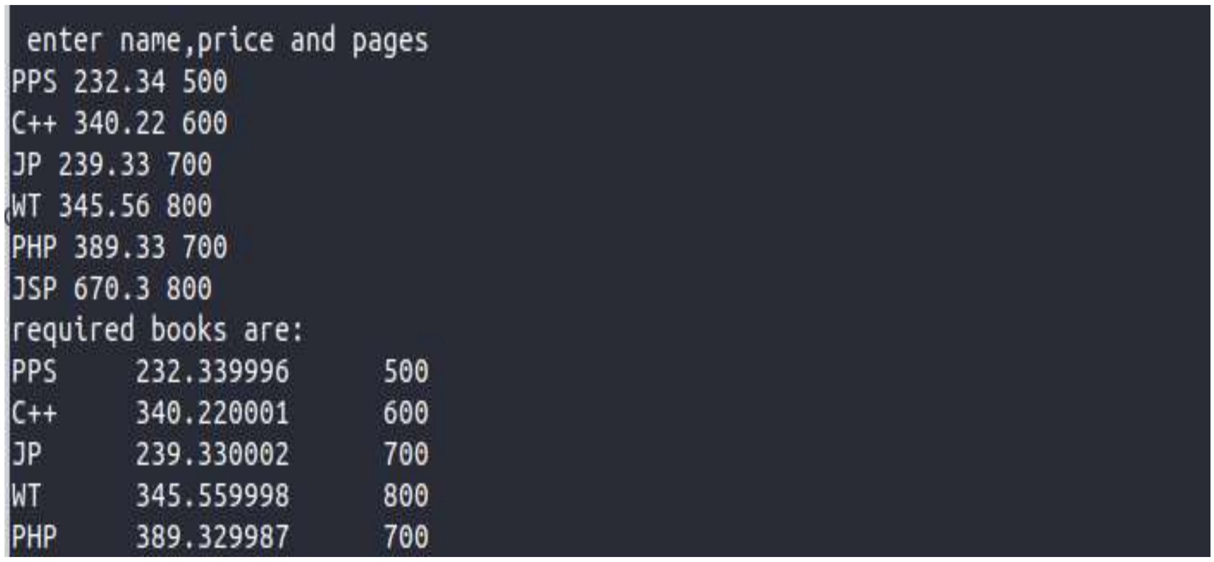
An array of structres in C can be defined as the collection of multiple structures variables where each variable contains information about different entities. The array of structures in C are used to store information about multiple entities of different data types.

Program:



```c
#include<stdio.h>
struct book{
        char name[20];
        float price;
        int pages;
};
int main(){
        struct book b[6];
        int i;
        printf("\n enter name,price and pages\n");
        for(i=0;i<6;i++){
                scanf("%s\t%f\t%d",b[i].name,&b[i].price,&b[i].pages);
        }
        printf("required books are:");
        for(i=0;i<6;i++){
                printf("\n%s\t%f\t%d",b[i].name,b[i].price,b[i].pages);
        }
        return 0;
}
```

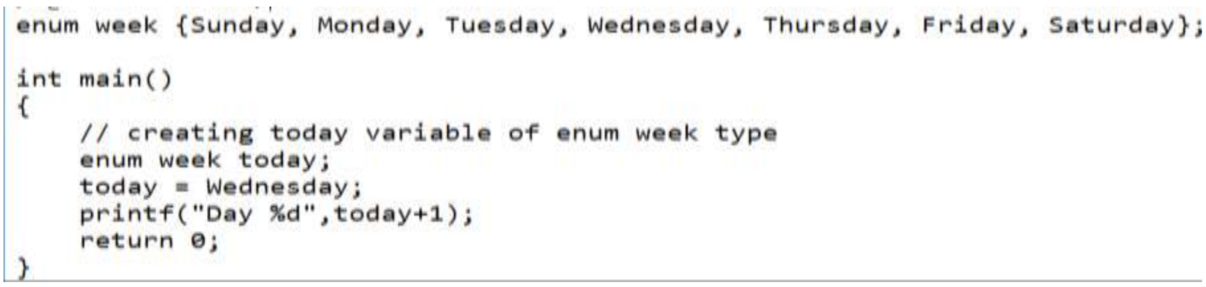
Output:



```
 enter name,price and pages
PPS 232.34 500
C++ 340.22 600
JP 239.33 700
WT 345.56 800
PHP 389.33 700
JSP 670.3 800
required books are:
PPS     232.339996      500
C++     340.220001      600
JP      239.330002      700
WT      345.559998      800
PHP     389.329987      700
```

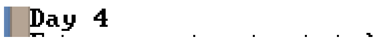
13. Write a c program to implement enumeration data type?

Program:



```
enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

int main()
{
    // creating today variable of enum week type
    enum week today;
    today = Wednesday;
    printf("Day %d",today+1);
    return 0;
}
```

Output:



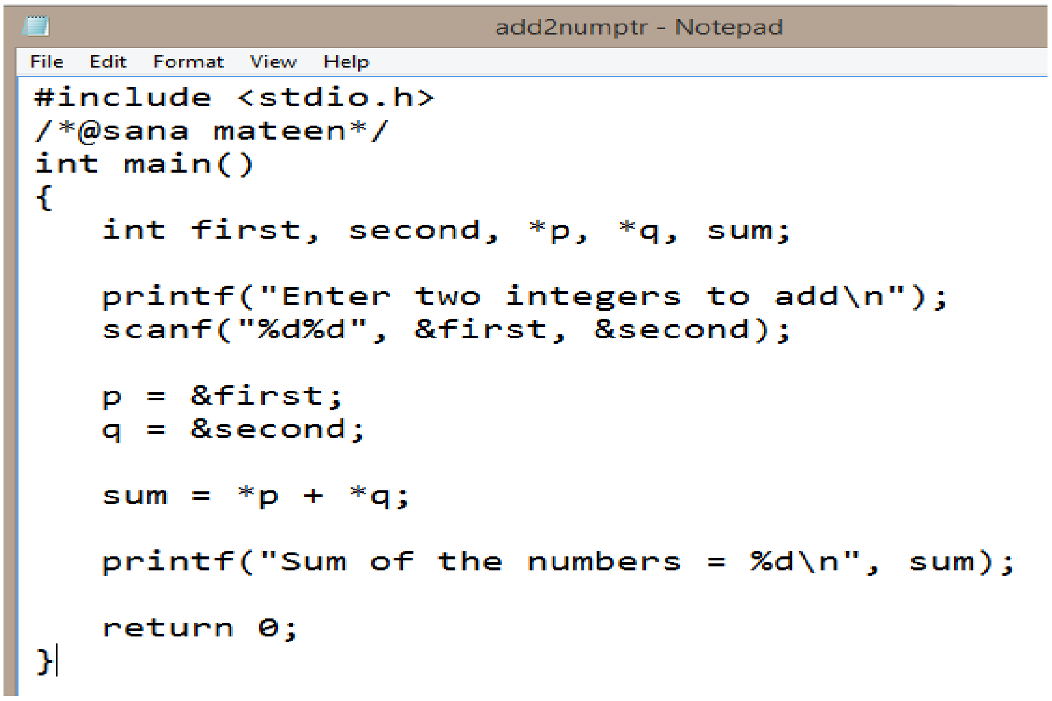
```
Day 4
```

14. Define pointer?Write a c program to perform addition of two numbers using pointers?

A **pointer** is a variable that stores the address of another variable.

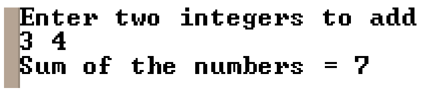
For example, an integer variable holds (or you can say stores) an integer value, however an integer pointer holds the address of a integer variable

Program:



```
#include <stdio.h>
/*@sana mateen*/
int main()
{
   int first, second, *p, *q, sum;

   printf("Enter two integers to add\n");
   scanf("%d%d", &first, &second);

   p = &first;
   q = &second;

   sum = *p + *q;

   printf("Sum of the numbers = %d\n", sum);

   return 0;
}
```

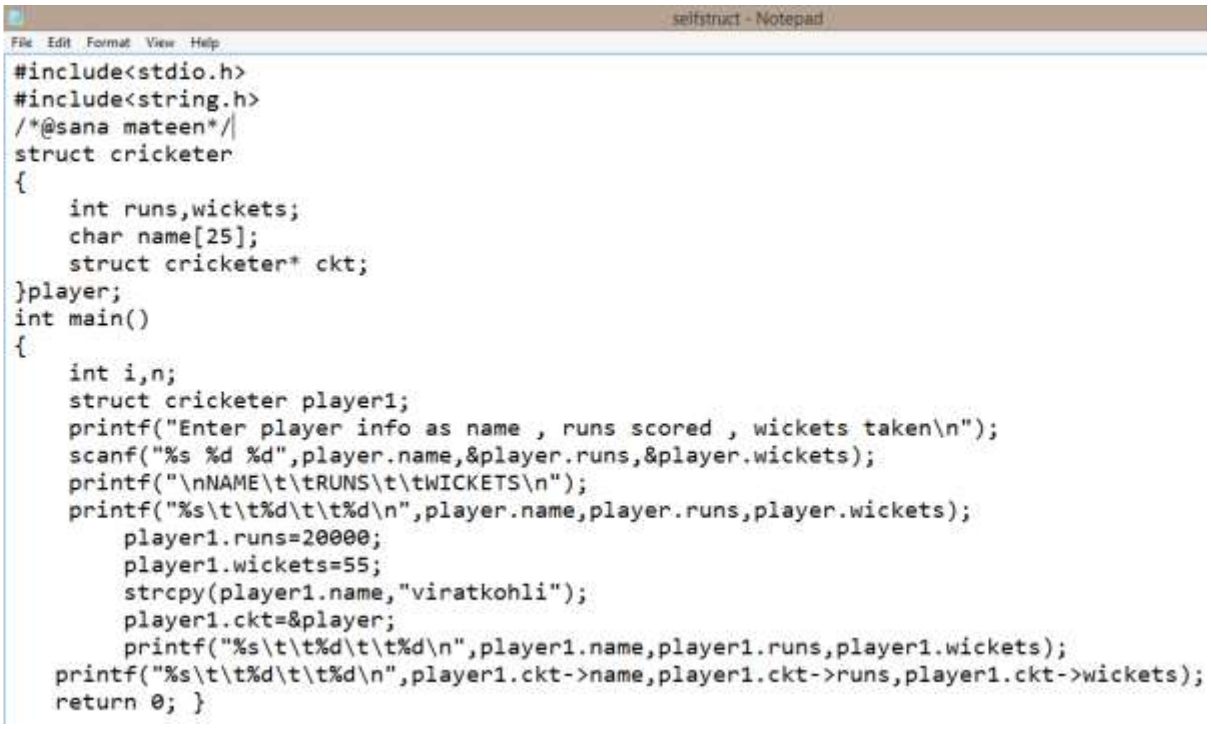
Output:



```
Enter two integers to add
3 4
Sum of the numbers = 7
```

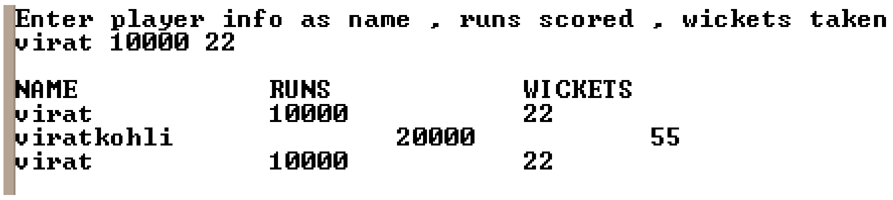
15. Write a c program to implement self referential structures?

Program:



```c
#include<stdio.h>
#include<string.h>
/*@sana mateen*/
struct cricketer
{
    int runs,wickets;
    char name[25];
    struct cricketer* ckt;
}player;
int main()
{
    int i,n;
    struct cricketer player1;
    printf("Enter player info as name , runs scored , wickets taken\n");
    scanf("%s %d %d",player.name,&player.runs,&player.wickets);
    printf("\nNAME\t\tRUNS\t\tWICKETS\n");
    printf("%s\t\t%d\t\t%d\n",player.name,player.runs,player.wickets);
        player1.runs=20000;
        player1.wickets=55;
        strcpy(player1.name,"viratkohli");
        player1.ckt=&player;
        printf("%s\t\t%d\t\t%d\n",player1.name,player1.runs,player1.wickets);
   printf("%s\t\t%d\t\t%d\n",player1.ckt->name,player1.ckt->runs,player1.ckt->wickets);
   return 0; }
```

Output:



```
Enter player info as name , runs scored , wickets taken
virat 10000 22

NAME            RUNS            WICKETS
virat           10000           22
viratkohli              20000           55
virat           10000           22
```

UNIT-3

Short Answer Questions for 2 Marks

1. What is preprocessor directive?Mention its usage?

Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing.

Commands used in preprocessor are called preprocessor directives and they begin with "#" symbol.

Preprocessor directives, such as #define and #ifdef , are typically used to make source programs easy to change and easy to compile in different execution environments.

2. Write about the following i)#include ii)#define iii)#undef

i)#include

The #include preprocessor directive is used to paste code of given file into current file. It is used include system-defined and user-defined header files. If included file is not found, compiler renders error.

ii)#define

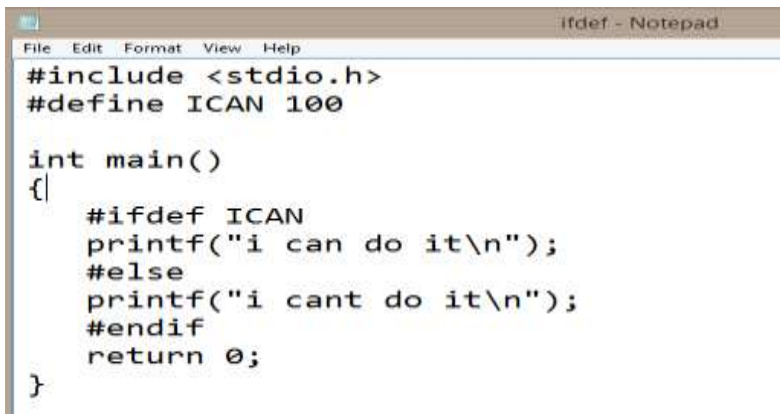
This macro defines constant value and can be any of the basic data types.The object-like macro is an identifier that is replaced by value.

iii)#undef

To undefine a macro means to cancel its definition. This is done with the #undef directive.

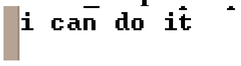
3. Write a c program to implement #ifdef directive?

Program:



```
#include <stdio.h>
#define ICAN 100

int main()
{
   #ifdef ICAN
   printf("i can do it\n");
   #else
   printf("i cant do it\n");
   #endif
   return 0;
}
```

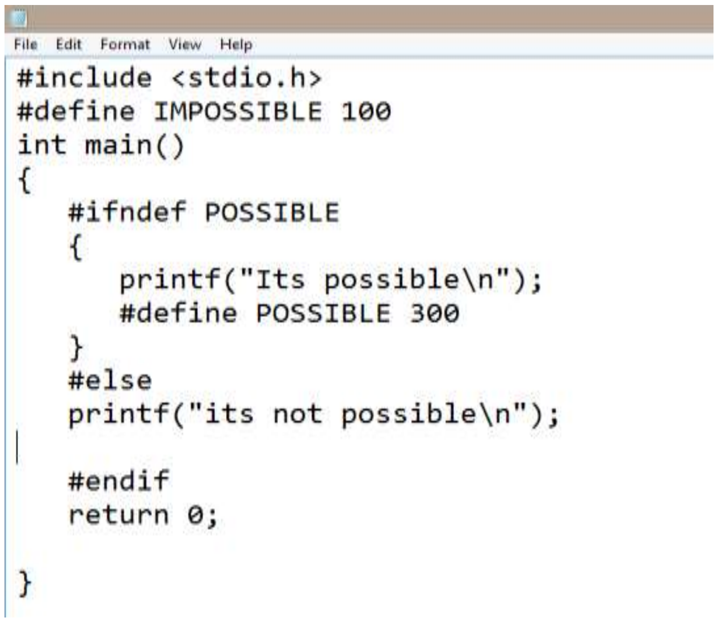
Output:



```
i can do it
```

4. Write a c program to implement #ifndef directive?

Program:

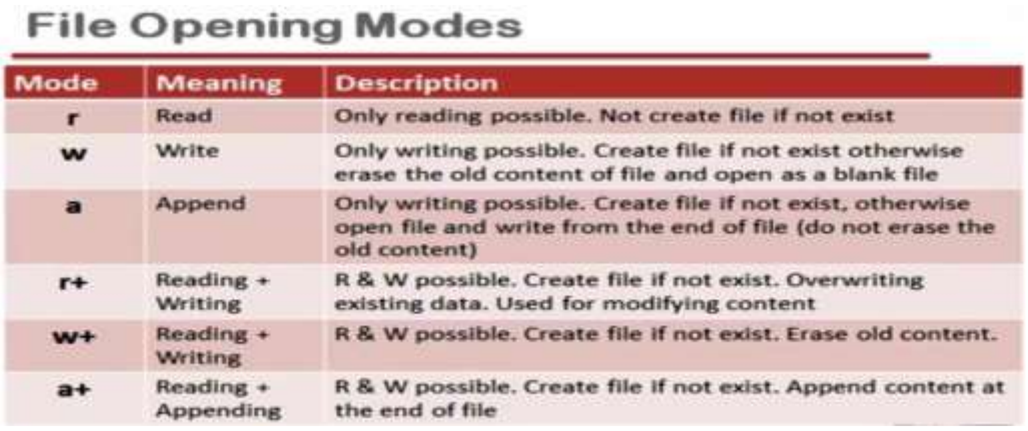


```
#include <stdio.h>
#define IMPOSSIBLE 100
int main()
{
   #ifndef POSSIBLE
   {
      printf("Its possible\n");
      #define POSSIBLE 300
   }
   #else
   printf("its not possible\n");

   #endif
   return 0;

}
```

5. **Differentiate between text and binary files?**

Text File	Binary File
A text file stores data in the form of alphabets, digits and other special symbols by storing their ASCII values and are in a human readable format. For example, any file with a .txt, .c, etc extension.	a binary file contains a sequence or a collection of bytes which are not in a human readable format. For example, files with .exe, .mp3, etc extension. It represents custom data.
A small error in a textual file can be recognized and eliminated when seen	Whereas, a small error in a binary file corrupts the file and is not easy to detect.

6. **What are the different modes of operations on text file?**

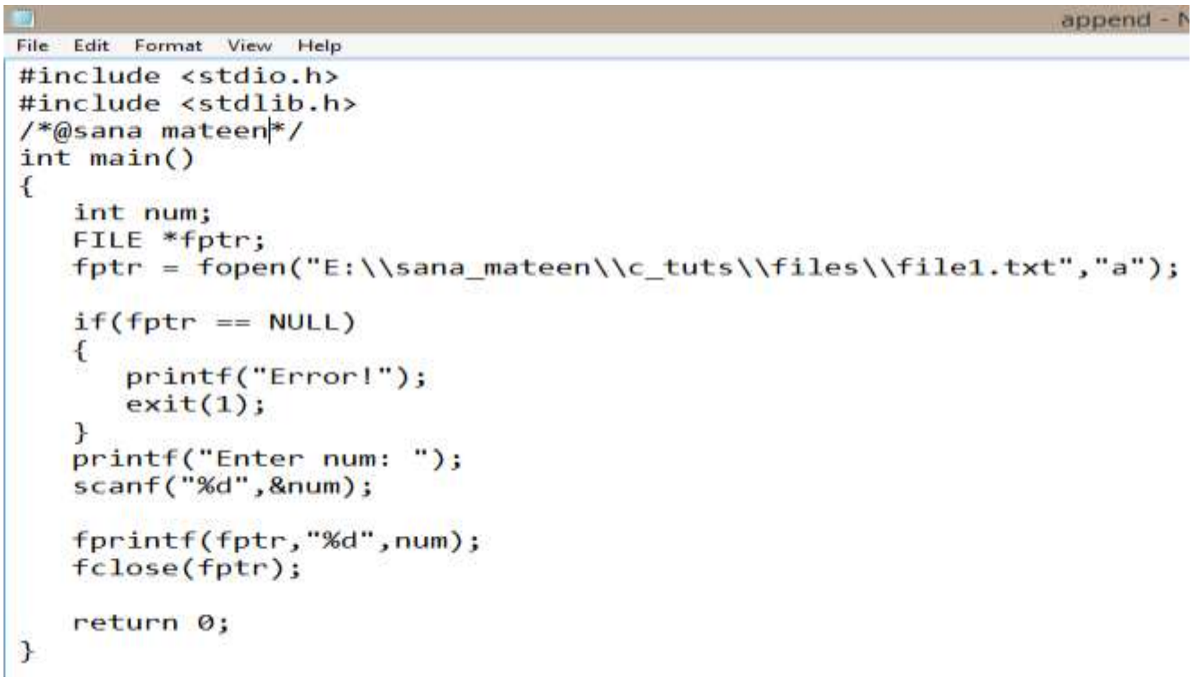


File Opening Modes

Mode	Meaning	Description
r	Read	Only reading possible. Not create file if not exist
w	Write	Only writing possible. Create file if not exist otherwise erase the old content of file and open as a blank file
a	Append	Only writing possible. Create file if not exist, otherwise open file and write from the end of file (do not erase the old content)
r+	Reading + Writing	R & W possible. Create file if not exist. Overwriting existing data. Used for modifying content
w+	Reading + Writing	R & W possible. Create file if not exist. Erase old content.
a+	Reading + Appending	R & W possible. Create file if not exist. Append content at the end of file

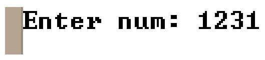
7. **Write a c program to append data to existing file?**

Program:



```
#include <stdio.h>
#include <stdlib.h>
/*@sana mateen*/
int main()
{
    int num;
    FILE *fptr;
    fptr = fopen("E:\\sana_mateen\\c_tuts\\files\\file1.txt","a");

    if(fptr == NULL)
    {
       printf("Error!");
       exit(1);
    }
    printf("Enter num: ");
    scanf("%d",&num);

    fprintf(fptr,"%d",num);
    fclose(fptr);

    return 0;
}
```

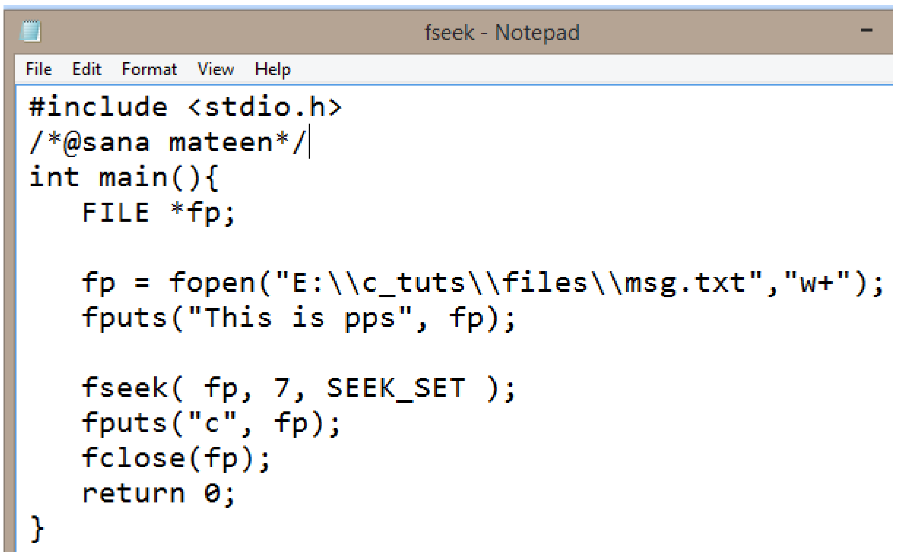
Output:



```
Enter num: 1231
```

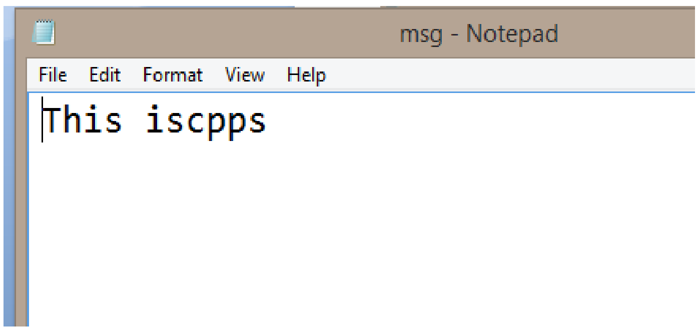
8. Write a c program to implement fseek()?

Program:



```
#include <stdio.h>
/*@sana mateen*/
int main(){
   FILE *fp;

   fp = fopen("E:\\c_tuts\\files\\msg.txt","w+");
   fputs("This is pps", fp);

   fseek( fp, 7, SEEK_SET );
   fputs("c", fp);
   fclose(fp);
   return 0;
}
```

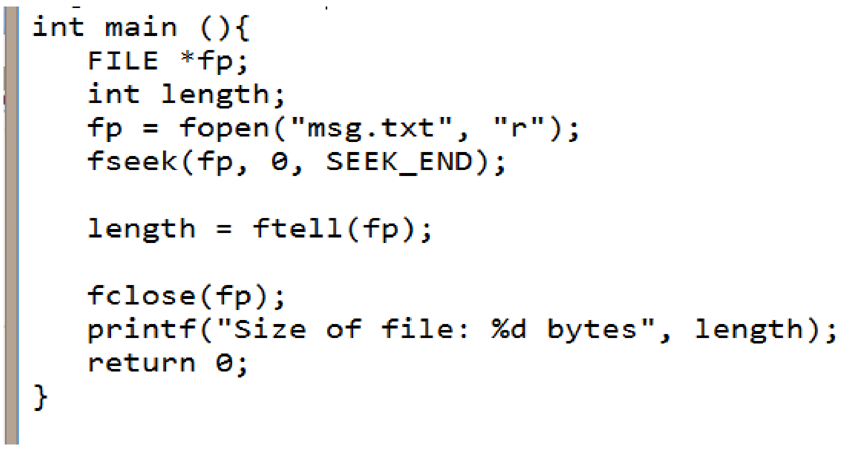
Output:



```
This iscpps
```

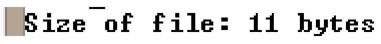
9. Write a c program to implement ftell()?

Program:



```
int main (){
    FILE *fp;
    int length;
    fp = fopen("msg.txt", "r");
    fseek(fp, 0, SEEK_END);

    length = ftell(fp);

    fclose(fp);
    printf("Size of file: %d bytes", length);
    return 0;
}
```

Output:

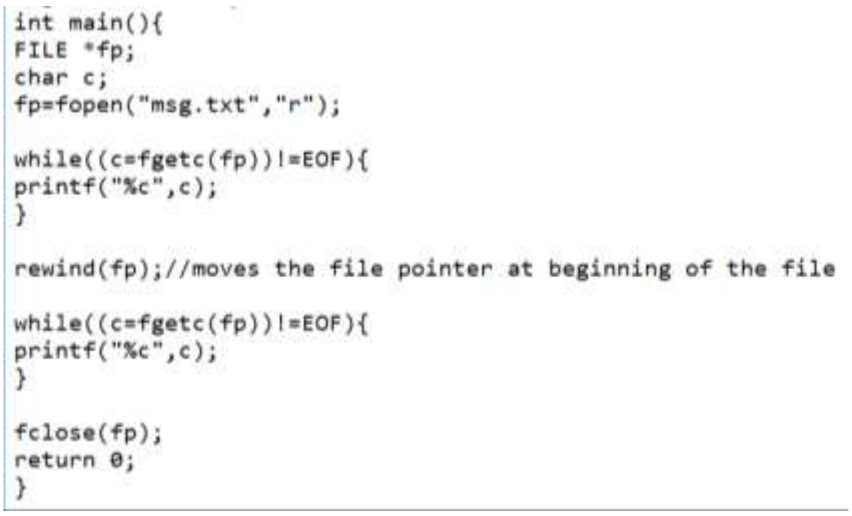


```
Size of file: 11 bytes
```

Short Answer Questions for 3 Marks:

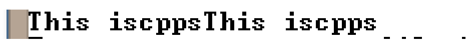
10. Write a c program to implement rewind()?

Program:



```
int main(){
FILE *fp;
char c;
fp=fopen("msg.txt","r");

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

rewind(fp);//moves the file pointer at beginning of the file

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

fclose(fp);
return 0;
}
```

Output:



```
This iscppsThis iscpps
```

11. Write the syntax of functions used in text and binary file operations?

Declaration: int fprintf(FILE *fp, const char *format, …)

fprintf() function is used to write formatted data into a file. In a C program, we use fprinf() as below.

fprintf (fp, "%s %d", "var1", var2);

Declaration: int fscanf(FILE *fp, const char *format, …)

fscanf() function is used to read formatted data from a file. In a C program, we use fscanf() as below.

fscanf(fp,"%d",&age);

The fwrite() function is used to write records (sequence of bytes) to the file. A record may be an array or a structure.

Syntax of fwrite() function

fwrite(ptr, int size, int n, FILE *fp);

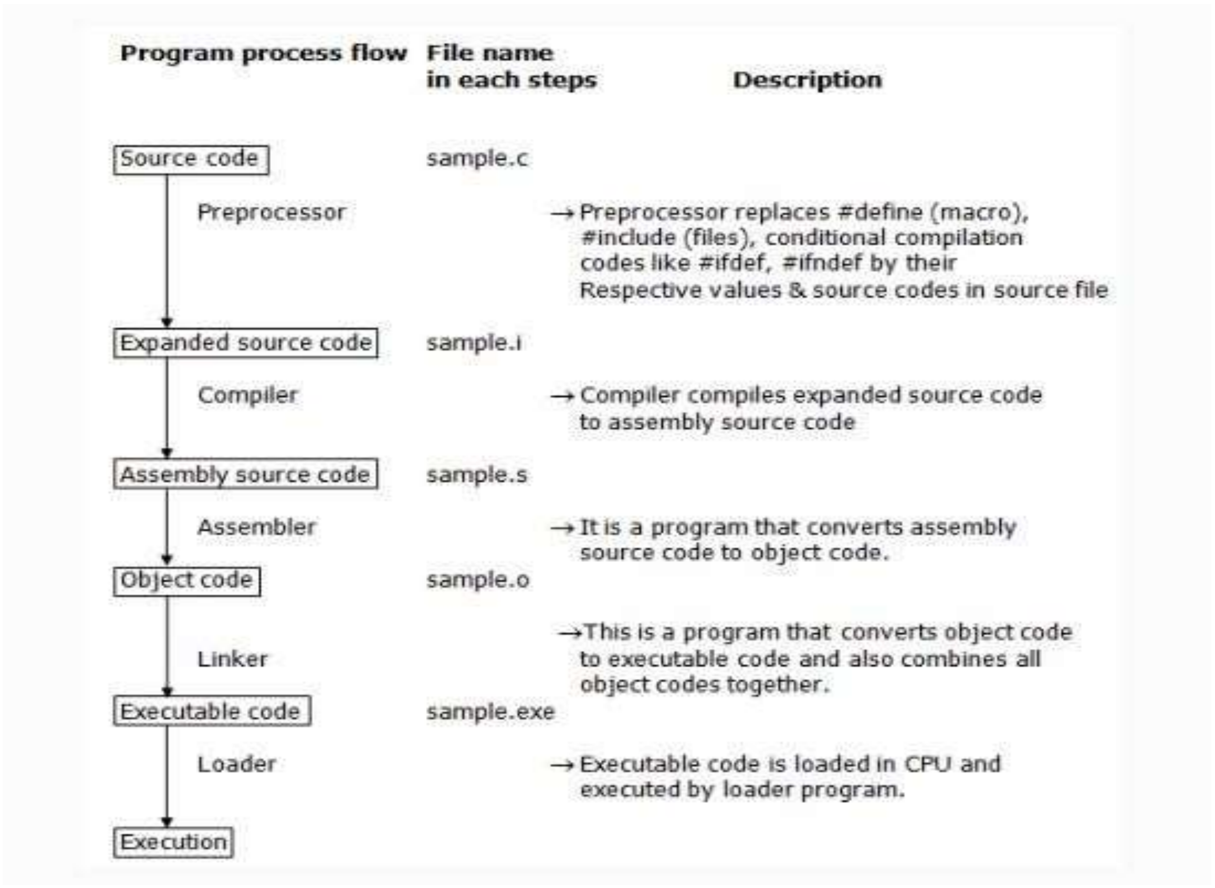
The fread() function is used to read bytes form the file.

Syntax of fread() function

fread(ptr, int size, int n, FILE *fp);

Long Answer Questions for 5 Marks:

1. **With a neat diagram explain the program process flow with each steps?**



Program process flow	File name in each steps	Description
Source code	sample.c	
Preprocessor		→ Preprocessor replaces #define (macro), #include (files), conditional compilation codes like #ifdef, #ifndef by their Respective values & source codes in source file
Expanded source code	sample.i	
Compiler		→ Compiler compiles expanded source code to assembly source code
Assembly source code	sample.s	
Assembler		→ It is a program that converts assembly source code to object code.
Object code	sample.o	
Linker		→ This is a program that converts object code to executable code and also combines all object codes together.
Executable code	sample.exe	
Loader		→ Executable code is loaded in CPU and executed by loader program.
Execution		

1) C program (source code) is sent to preprocessor first. The preprocessor is responsible to convert preprocessor directives into their respective values. The preprocessor generates an expanded source code.

2) Expanded source code is sent to compiler which compiles the code and converts it into assembly code.

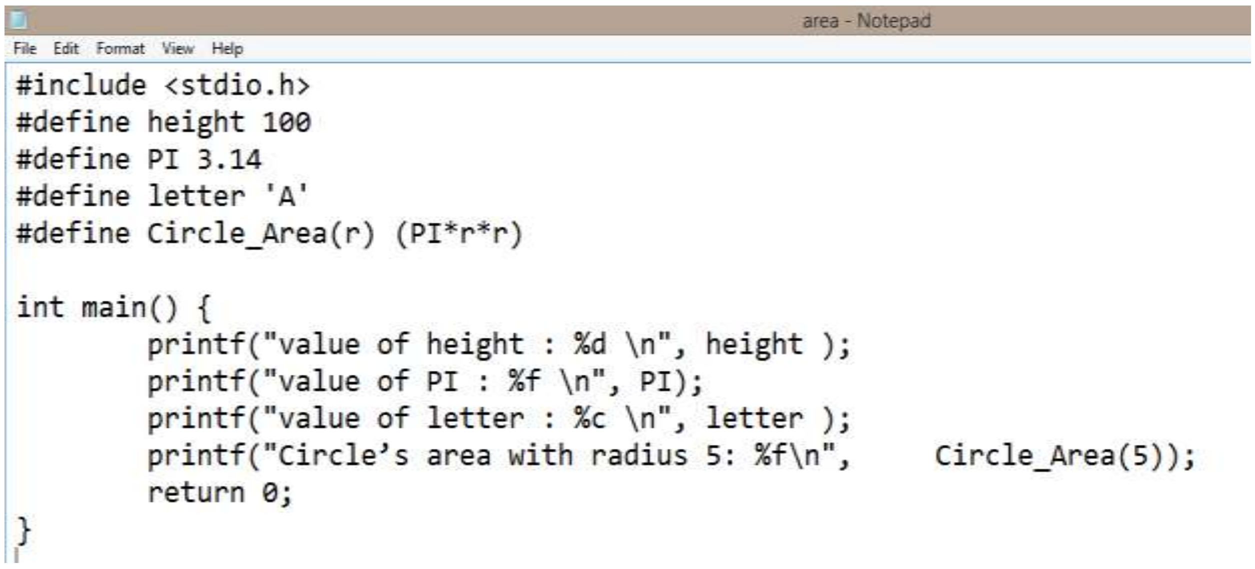
3) The assembly code is sent to assembler which assembles the code and converts it into object code. Now a simple.obj file is generated.

4) The object code is sent to linker which links it to the library such as header files. Then it is converted into executable code. A simple.exe file is generated.

5) The executable code is sent to loader which loads it into memory and then it is executed. After execution, output is sent to console.

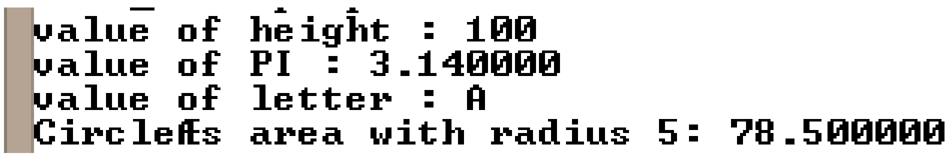
2. Write a c program to find area of circle using #define preprocessor?

Program:



```c
#include <stdio.h>
#define height 100
#define PI 3.14
#define letter 'A'
#define Circle_Area(r) (PI*r*r)

int main() {
        printf("value of height : %d \n", height );
        printf("value of PI : %f \n", PI);
        printf("value of letter : %c \n", letter );
        printf("Circle’s area with radius 5: %f\n",     Circle_Area(5));
        return 0;
}
```

Output:



```
value of height : 100
value of PI : 3.140000
value of letter : A
Circleÿs area with radius 5: 78.500000
```

3. Explain #define preprocessor directive in detail?

#define – This macro defines constant value and can be any of the basic data types.

The object-like macro is an identifier that is replaced by value.

It is widely used to represent numeric constants. For example:

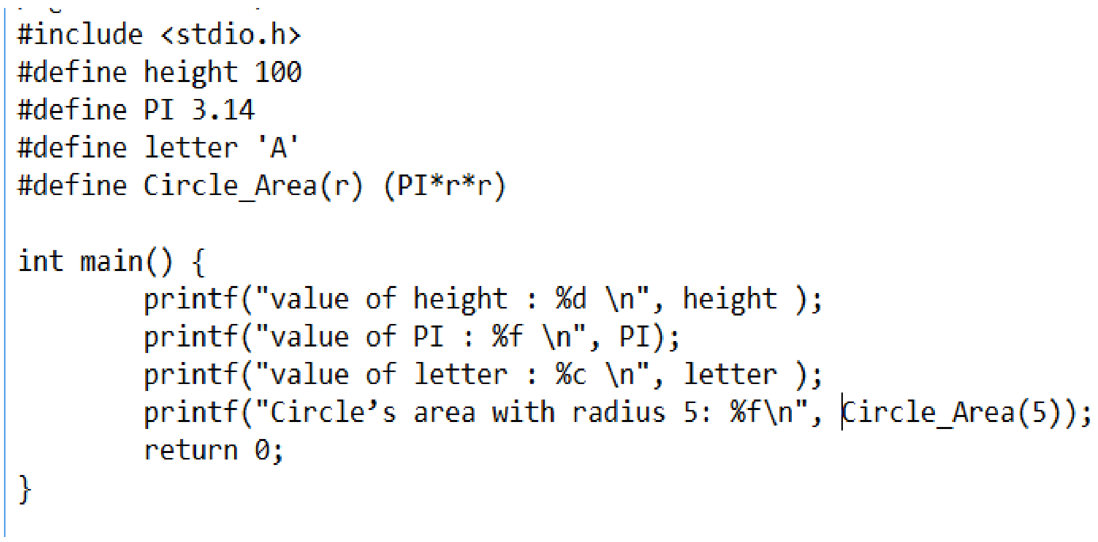
#define PI 3.1415

The function-like macro looks like function call. For example:

#define MIN(a,b) ((a)<(b)?(a):(b))

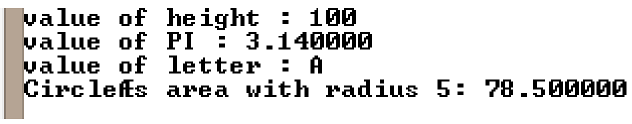
Here, MIN is the macro name. Let's see an example of Function-like Macros

Program:



```
#include <stdio.h>
#define height 100
#define PI 3.14
#define letter 'A'
#define Circle_Area(r) (PI*r*r)

int main() {
        printf("value of height : %d \n", height );
        printf("value of PI : %f \n", PI);
        printf("value of letter : %c \n", letter );
        printf("Circle’s area with radius 5: %f\n", Circle_Area(5));
        return 0;
}
```

Output:



```
value of height : 100
value of PI : 3.140000
value of letter : A
CircleÆs area with radius 5: 78.500000
```

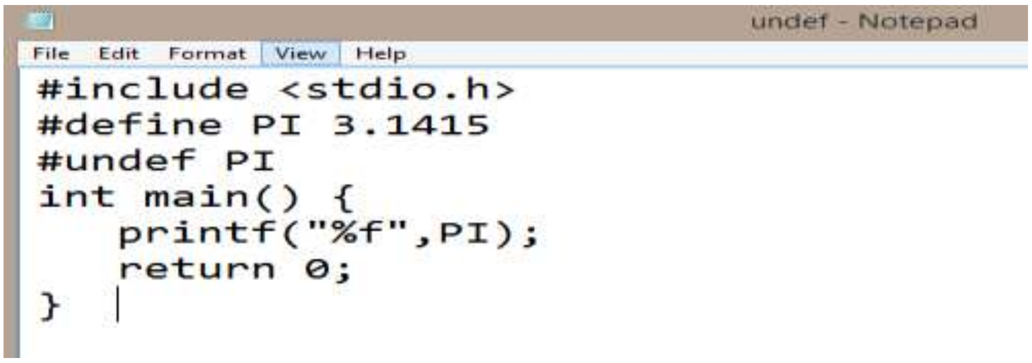
4. Write a c program to implement #undef directive?

To undefine a macro means to cancel its definition. This is done with the #undef directive.

Syntax:

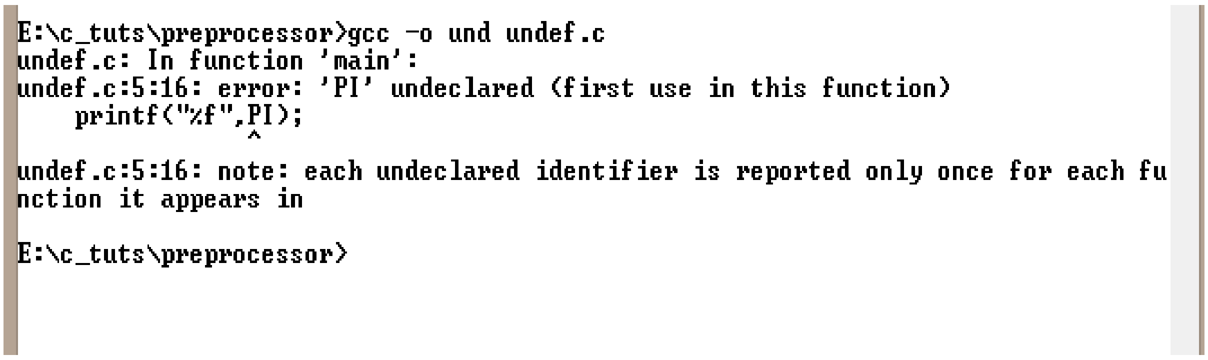
#undef token

Program:



```
#include <stdio.h>
#define PI 3.1415
#undef PI
int main() {
    printf("%f",PI);
    return 0;
}
```

Output:



```
E:\c_tuts\preprocessor>gcc -o und undef.c
undef.c: In function 'main':
undef.c:5:16: error: 'PI' undeclared (first use in this function)
    printf("%f",PI);
                ^
undef.c:5:16: note: each undeclared identifier is reported only once for each fu
nction it appears in

E:\c_tuts\preprocessor>
```

5. Explain the following directives along with an example program i)#ifdef ii)#ifndef iii)#if iv)#else

i)#ifdef

The #ifdef preprocessor directive checks if macro is defined by #define. If yes, it executes the code.

Syntax:

#ifdef MACRO

//code

#endif

ii)#ifndef

The #ifndef preprocessor directive checks if macro is not defined by #define. If yes, it executes the code.

Syntax:

```
#ifndef MACRO
 //code
#endif
```

iii)#if

The #if preprocessor directive evaluates the expression or condition. If condition is true, it executes the code.

Syntax:

```
#if expression
//code
 #endif
```

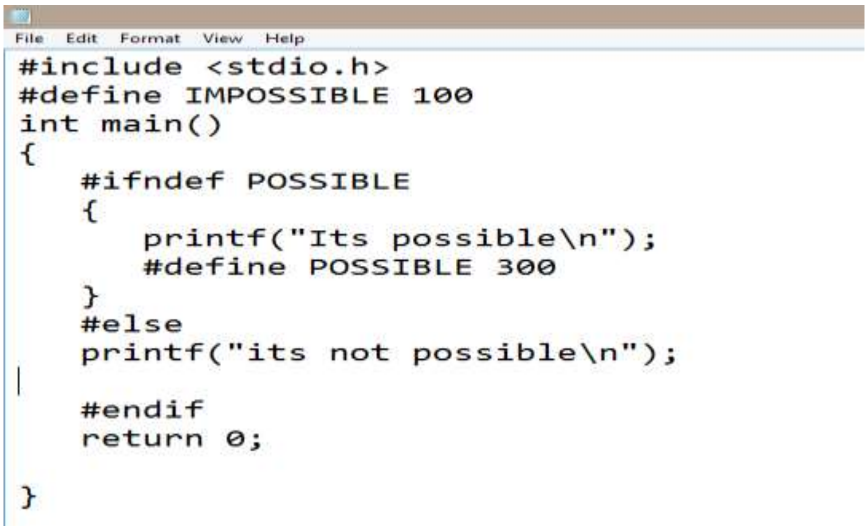
iv) #else

The #else preprocessor directive evaluates the expression or condition if condition of #if is false. It can be used with #if, #elif, #ifdef and #ifndef directives.

Syntax:

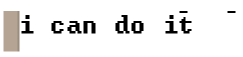
```
#if expression
//if code
#else
//else code
#endif
```

Program:



```
#include <stdio.h>
#define IMPOSSIBLE 100
int main()
{
   #ifndef POSSIBLE
   {
      printf("Its possible\n");
      #define POSSIBLE 300
   }
   #else
   printf("its not possible\n");

   #endif
   return 0;

}
```

Output:



```
i can do it
```

6. Explain the following directives along with an example program i)#elif

#if expression

//if code

#elif expression

//elif code

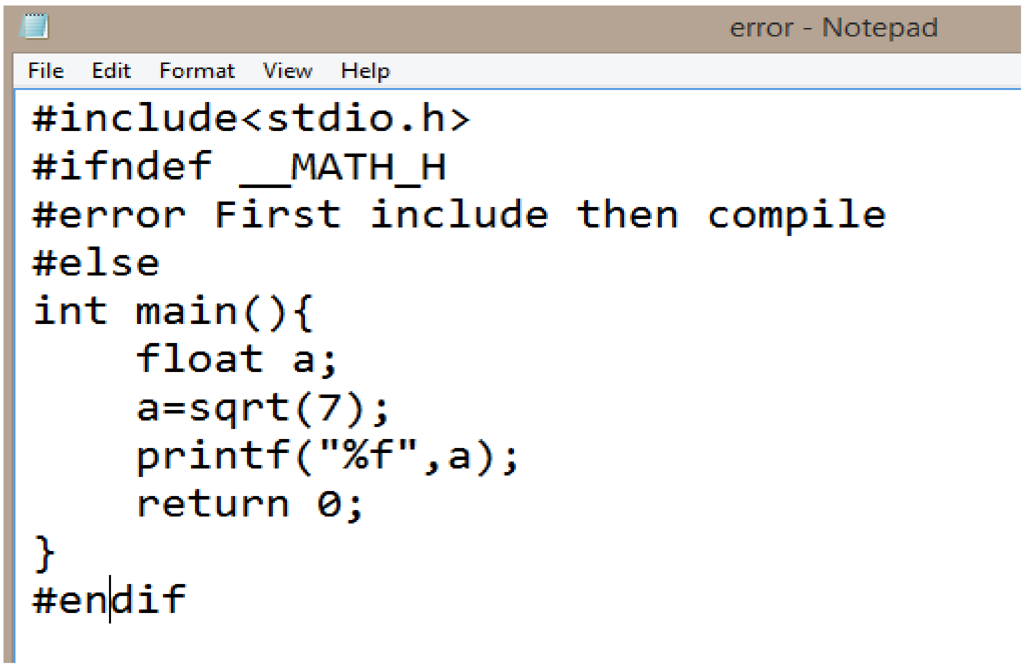
#else

//else code

#endif

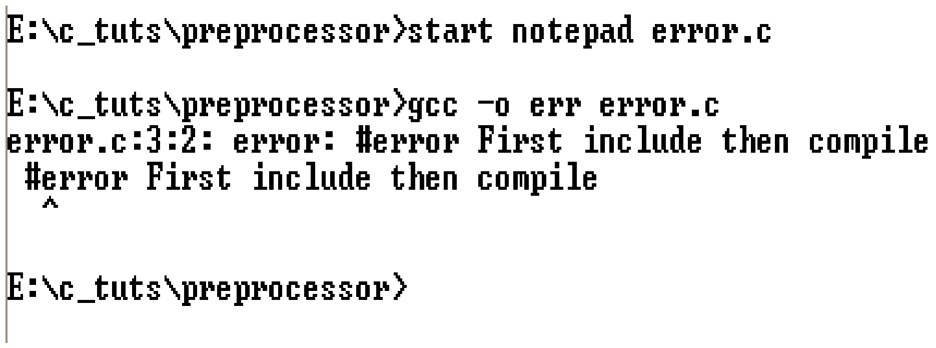
ii)#error

The #error preprocessor directive indicates error. The compiler gives fatal error if #error directive is found and skips further compilation process.



```
#include<stdio.h>
#ifndef __MATH_H
#error First include then compile
#else
int main(){
    float a;
    a=sqrt(7);
    printf("%f",a);
    return 0;
}
#endif
```

Output:



```
E:\c_tuts\preprocessor>start notepad error.c

E:\c_tuts\preprocessor>gcc -o err error.c
error.c:3:2: error: #error First include then compile
 #error First include then compile
  ^

E:\c_tuts\preprocessor>
```

iii)#pragma

The #pragma preprocessor directive is used to provide additional information to the compiler.

The #pragma directive is used by the compiler to offer machine or operating-system feature.

Different compilers can provide different usage of #pragma directive.

Syntax:

#pragma token

7. **What is a File?Explain the different types of Files along with their mode of operations?**

A file is a container in computer storage devices used for storing data.

Why files are needed?

When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

If you have to enter a large number of data, it will take a lot of time to enter them all.
However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.

You can easily move your data from one computer to another without any changes.

When dealing with files, there are two types of files you should know about:

Text files

Binary files

1. Text files

Text files are the normal **.txt** files. You can easily create text files using any simple text editors such as Notepad.

When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

They take minimum effort to maintain, are easily readable, and provide the least security and takes bigger storage space.

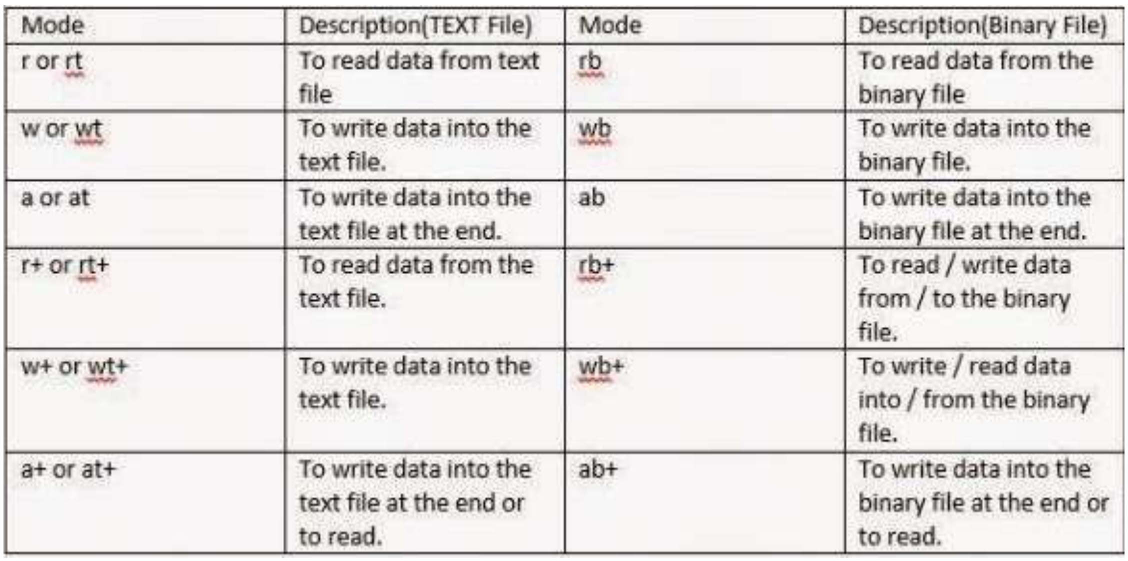
2. Binary files

Binary files are mostly the **.bin** files in your computer.

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

They can hold a higher amount of data, are not readable easily, and provides better security than text files.

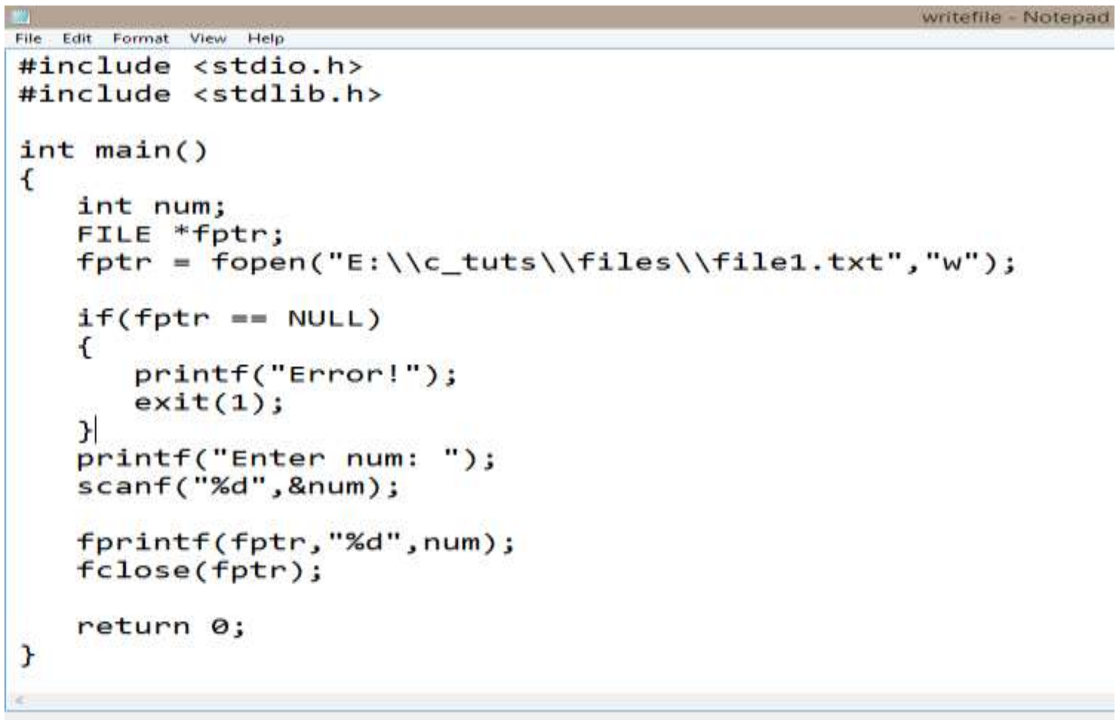
Modes of Text File and Binary File:



Mode	Description(TEXT File)	Mode	Description(Binary File)
r or rt	To read data from text file	rb	To read data from the binary file
w or wt	To write data into the text file.	wb	To write data into the binary file.
a or at	To write data into the text file at the end.	ab	To write data into the binary file at the end.
r+ or rt+	To read data from the text file.	rb+	To read / write data from / to the binary file.
w+ or wt+	To write data into the text file.	wb+	To write / read data into / from the binary file.
a+ or at+	To write data into the text file at the end or to read.	ab+	To write data into the binary file at the end or to read.

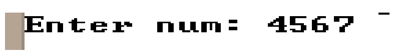
8. **Write a c program to read and write the data to text file using fprintf and fscanf function?**

Program:



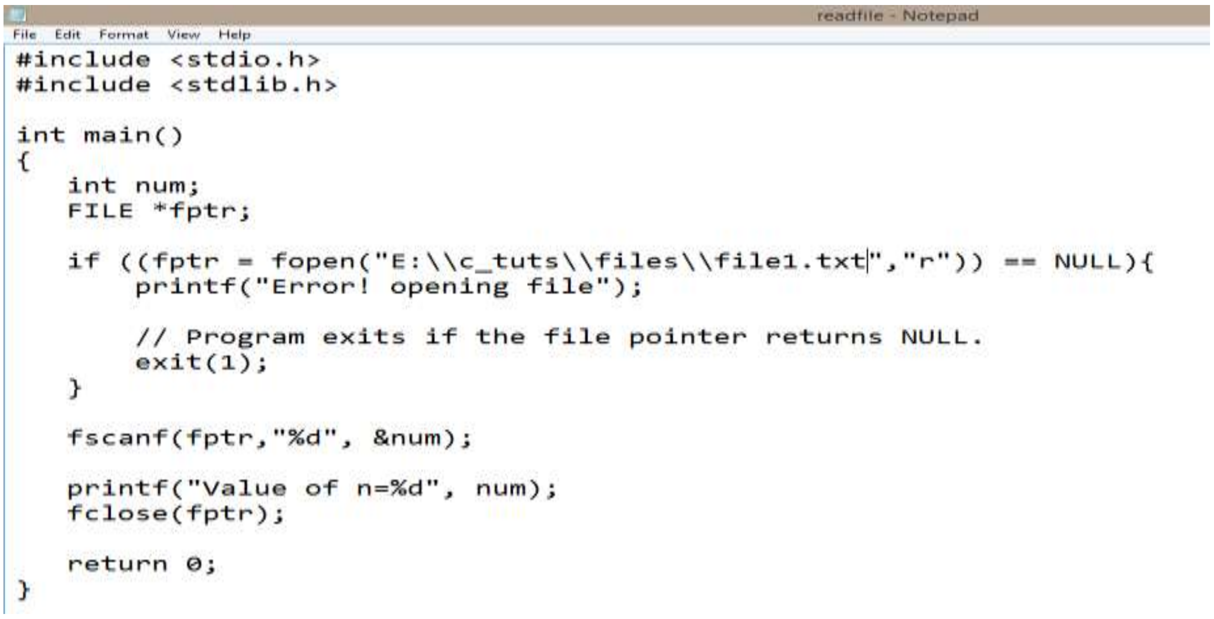
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num;
    FILE *fptr;
    fptr = fopen("E:\\c_tuts\\files\\file1.txt","w");

    if(fptr == NULL)
    {
       printf("Error!");
       exit(1);
    }
    printf("Enter num: ");
    scanf("%d",&num);

    fprintf(fptr,"%d",num);
    fclose(fptr);

    return 0;
}
```

Output:



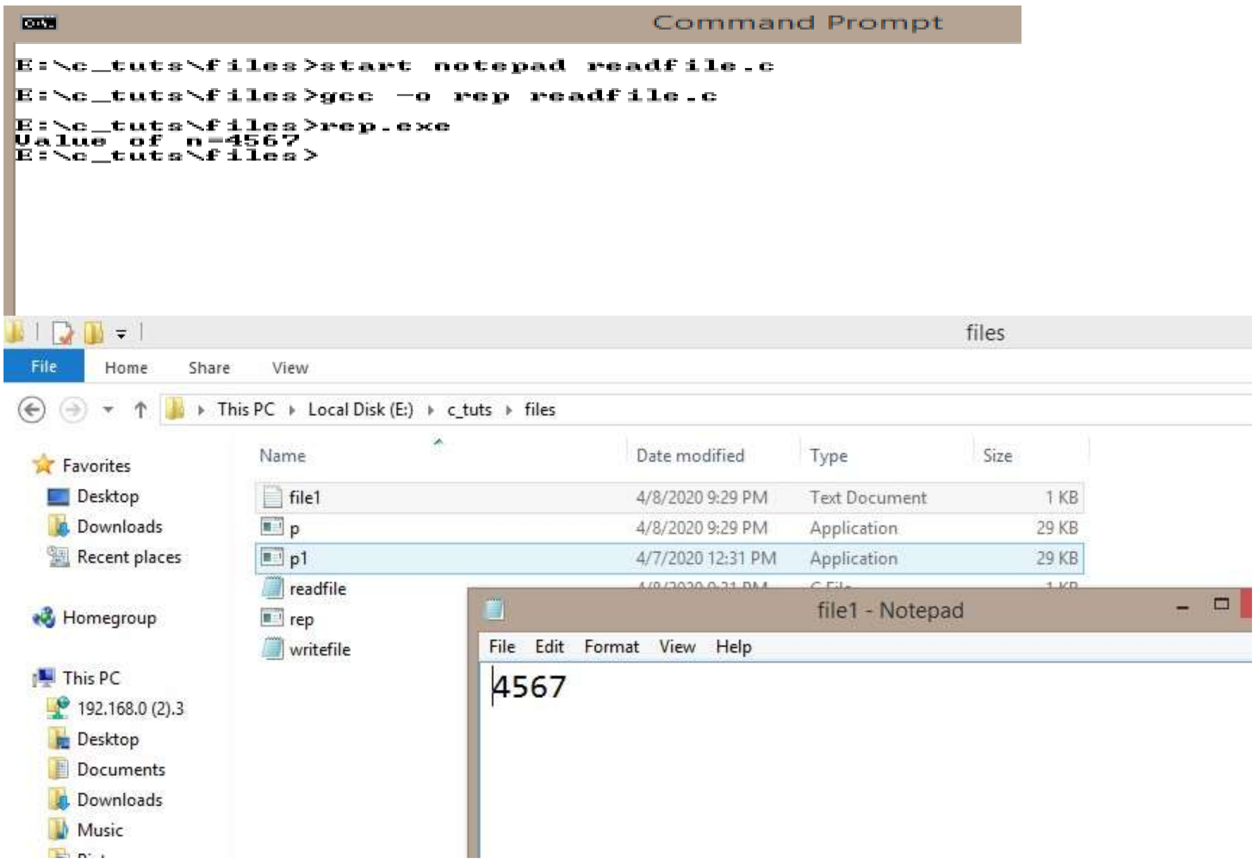
```
Enter num: 4567
```

Program:



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

int main()
{
    int num;
    FILE *fptr;

    if ((fptr = fopen("E:\\c_tuts\\files\\file1.txt","r")) == NULL){
        printf("Error! opening file");

        // Program exits if the file pointer returns NULL.
        exit(1);
    }

    fscanf(fptr,"%d", &num);

    printf("Value of n=%d", num);
    fclose(fptr);

    return 0;
}
```

Output:



```
E:\c_tuts\files>start notepad readfile.c

E:\c_tuts\files>gcc -o rep readfile.c

E:\c_tuts\files>rep.exe
Value of n=4567
E:\c_tuts\files>
```

Name	Date modified	Type	Size
file1	4/8/2020 9:29 PM	Text Document	1 KB
p	4/8/2020 9:29 PM	Application	29 KB
p1	4/7/2020 12:31 PM	Application	29 KB
readfile			
rep			
writefile			

```
4567
```

9. Explain the process of reading and writing information to binary file along with an example program?

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.

Writing to a binary file

To write into a binary file, you need to use the fwrite() function. The functions take four arguments:

address of data to be written in the disk

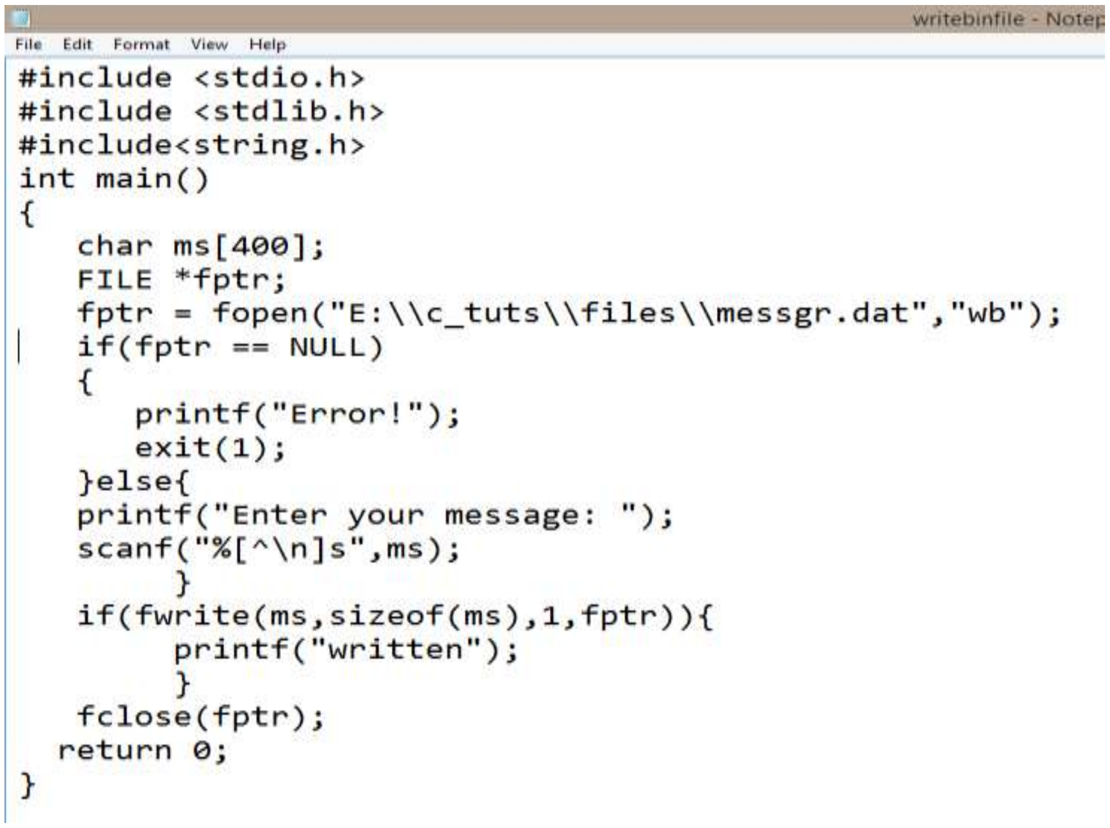
size of data to be written in the disk

number of such type of data

pointer to the file where you want to write.

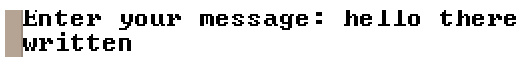
fwrite(addressData, sizeData, numbersData, pointerToFile);

Program:



```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
int main()
{
   char ms[400];
   FILE *fptr;
   fptr = fopen("E:\\c_tuts\\files\\messgr.dat","wb");
   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }else{
   printf("Enter your message: ");
   scanf("%[^\n]s",ms);
        }
   if(fwrite(ms,sizeof(ms),1,fptr)){
        printf("written");
        }
   fclose(fptr);
  return 0;
}
```

Output:



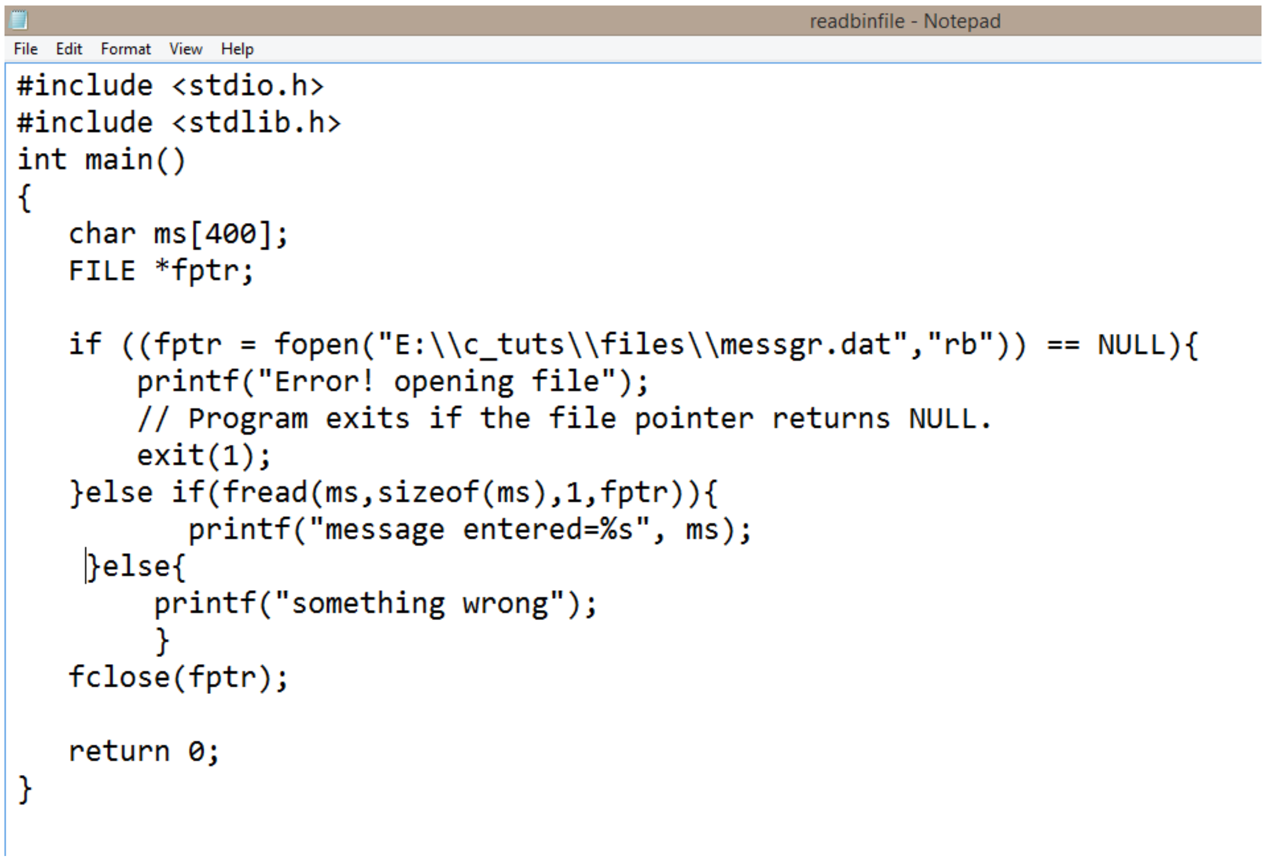
```
Enter your message: hello there
written
```

Function fread() also take 4 arguments similar to the fwrite() function

- address of data to be written in the disk
- size of data to be written in the disk
- number of such type of data
- pointer to the file where you want to write.

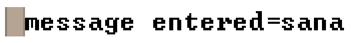
fread(addressData, sizeData, numbersData, pointerToFile);

Program:



```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char ms[400];
   FILE *fptr;

   if ((fptr = fopen("E:\\c_tuts\\files\\messgr.dat","rb")) == NULL){
       printf("Error! opening file");
       // Program exits if the file pointer returns NULL.
       exit(1);
   }else if(fread(ms,sizeof(ms),1,fptr)){
           printf("message entered=%s", ms);
    }else{
        printf("something wrong");
        }
   fclose(fptr);

   return 0;
}
```

Output:



```
message entered=sana
```

10. Explain fseek function in detail along with an example program?

The fseek() function is used to set the file pointer to the specified offset. It is used to write data into file at desired location.

Syntax:

int fseek(**FILE** *stream, **long int** offset, **int** whence)

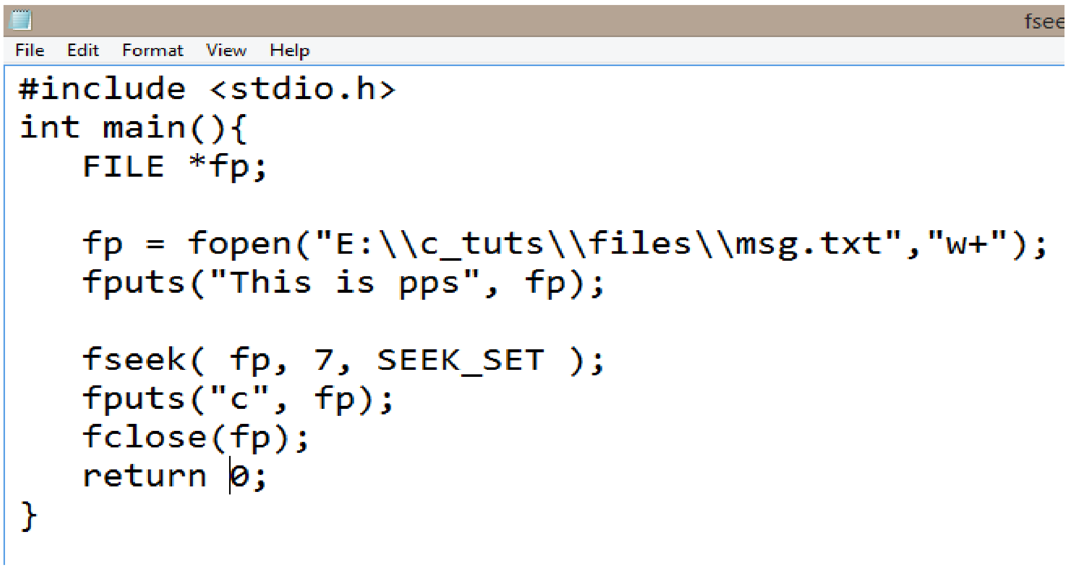
There are 3 constants used in the fseek() function for whence: SEEK_SET, SEEK_CUR and SEEK_END.

SEEK_SET – It moves file pointer position to the beginning of the file.

SEEK_CUR – It moves file pointer position to given location.

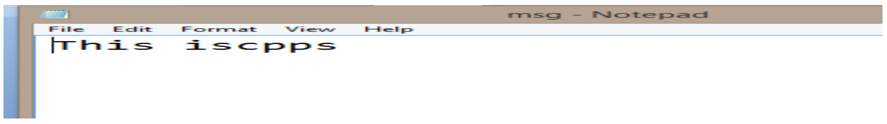
SEEK_END – It moves file pointer position to the end of file.

Program:



```
#include <stdio.h>
int main(){
   FILE *fp;

   fp = fopen("E:\\c_tuts\\files\\msg.txt","w+");
   fputs("This is pps", fp);

   fseek( fp, 7, SEEK_SET );
   fputs("c", fp);
   fclose(fp);
   return 0;
}
```

Output:



```
This iscpps
```

11. Explain ftell and rewind function through an example program?

The ftell() function returns the current file position of the specified stream.

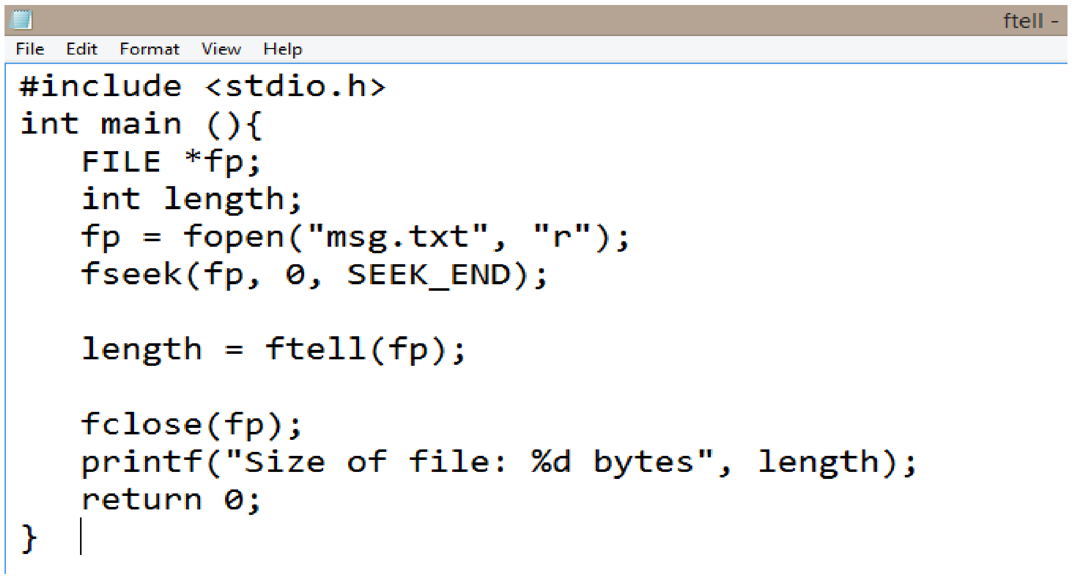
We can use ftell() function to get the total size of a file after moving file pointer at the end of file.

We can use. SEEK_END constant to move the file pointer at the end of file

Syntax:

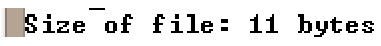
long int ftell(FILE *stream)

Program:



```
#include <stdio.h>
int main (){
   FILE *fp;
   int length;
   fp = fopen("msg.txt", "r");
   fseek(fp, 0, SEEK_END);

   length = ftell(fp);

   fclose(fp);
   printf("Size of file: %d bytes", length);
   return 0;
}
```

Output:



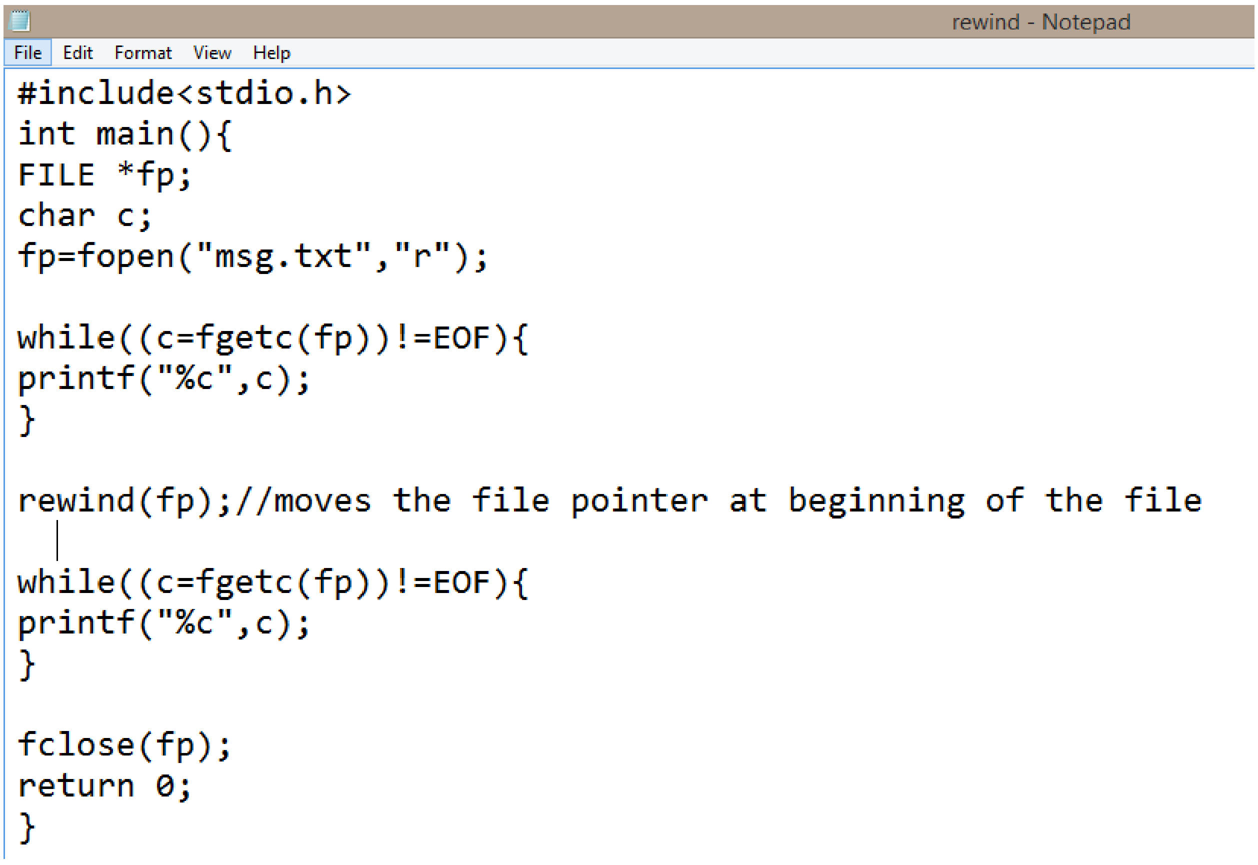
```
Size of file: 11 bytes
```

The rewind() function sets the file pointer at the beginning of the stream. It is useful if you have to use stream many times.

Syntax:

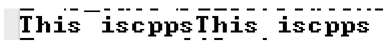
void rewind(FILE *stream)

Program:



```c
#include<stdio.h>
int main(){
FILE *fp;
char c;
fp=fopen("msg.txt","r");

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

rewind(fp);//moves the file pointer at beginning of the file

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}

fclose(fp);
return 0;
}
```

Output:



```
This iscppsThis iscpps
```

UNIT-4

Short Answer Questions for 2 Marks

1. **What is function?How to declare a function along with an example?**
   A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.
   A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.
   Defining a Function
   The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list ) {
  body of the function
}
int max(int num1, int num2) {

  /* local variable declaration */
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

2. **Differentiate between formal arguments and actual arguments?**

Formal Arguments	Actual Arguments
Arguments which are mentioned in the definition of the function is called formal arguments.	**Arguments which are mentioned in the function call is known as the actual argument. For example: func1(12, 23);**

**Formal arguments are very similar to local variables inside the function. Just like local variables, formal arguments are destroyed when the function ends.
int factorial(int n)
{ // write logic here
}**
	**Actual arguments can be constant, variables, expressions etc.
	func1(a, b); // here actual arguments are variable
	func1(a + b, b + a); // here actual arguments are expression**

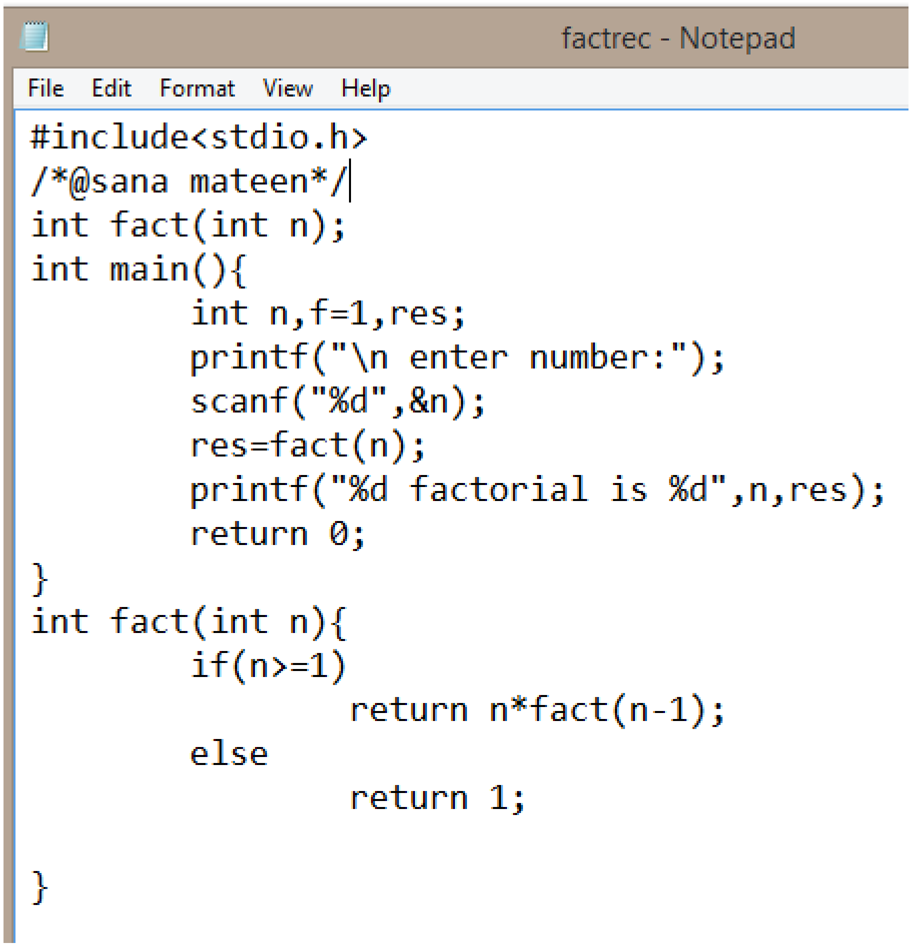
3. **Differentiate between call by value and call by reference?**

Call By Value	Call By Reference
In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.	In call by reference, the address of the variable is passed into the function call as the actual parameter.
In call by value method, we can not modify the value of the actual parameter by the formal parameter.	The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.	In call by reference, the memory allocation is similar for both formal parameters and actual parameters.
The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.	All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.

Short Answer Questions for 3 Marks:

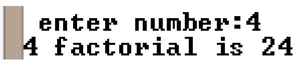
4. **Write a c program to find factorial of number using recursion?**

Program:



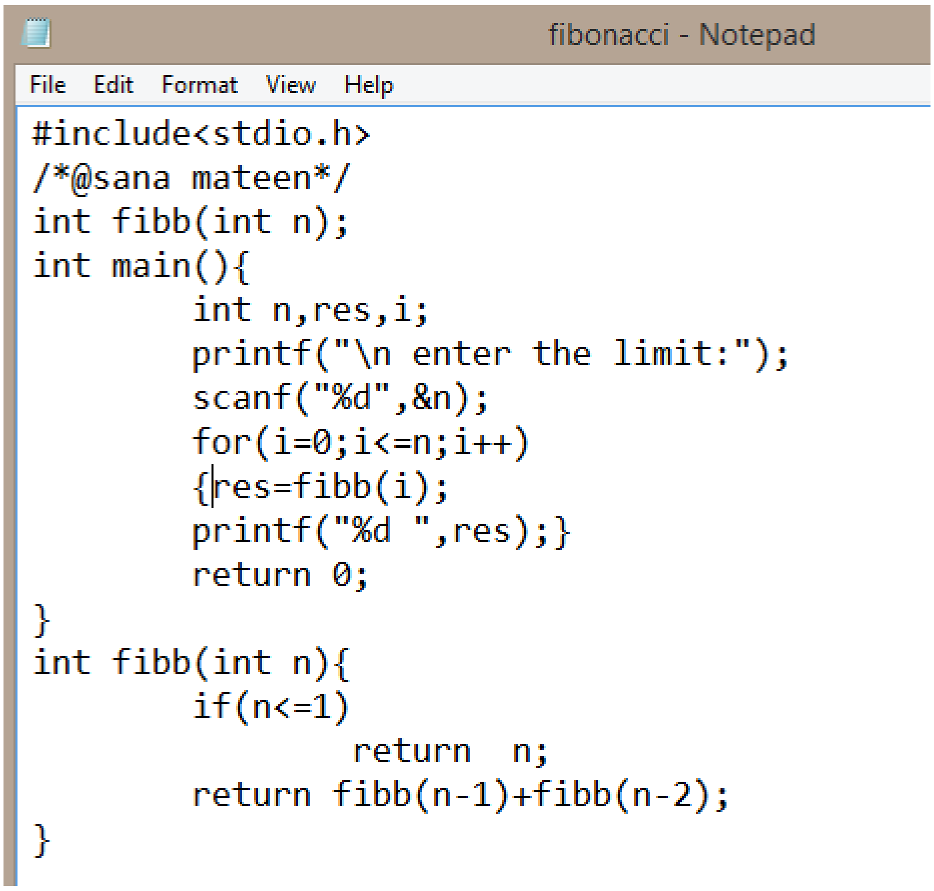
```
#include<stdio.h>
/*@sana mateen*/
int fact(int n);
int main(){
        int n,f=1,res;
        printf("\n enter number:");
        scanf("%d",&n);
        res=fact(n);
        printf("%d factorial is %d",n,res);
        return 0;
}
int fact(int n){
        if(n>=1)
                return n*fact(n-1);
        else
                return 1;

}
```

Output:



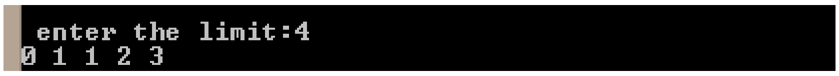
```
 enter number:4
4 factorial is 24
```

5. **Write a c program to find Fibonacci series using recursion?**
   Program:



```
#include<stdio.h>
/*@sana mateen*/
int fibb(int n);
int main(){
        int n,res,i;
        printf("\n enter the limit:");
        scanf("%d",&n);
        for(i=0;i<=n;i++)
        {res=fibb(i);
        printf("%d ",res);}
        return 0;
}
int fibb(int n){
        if(n<=1)
                return  n;
        return fibb(n-1)+fibb(n-2);
}
```

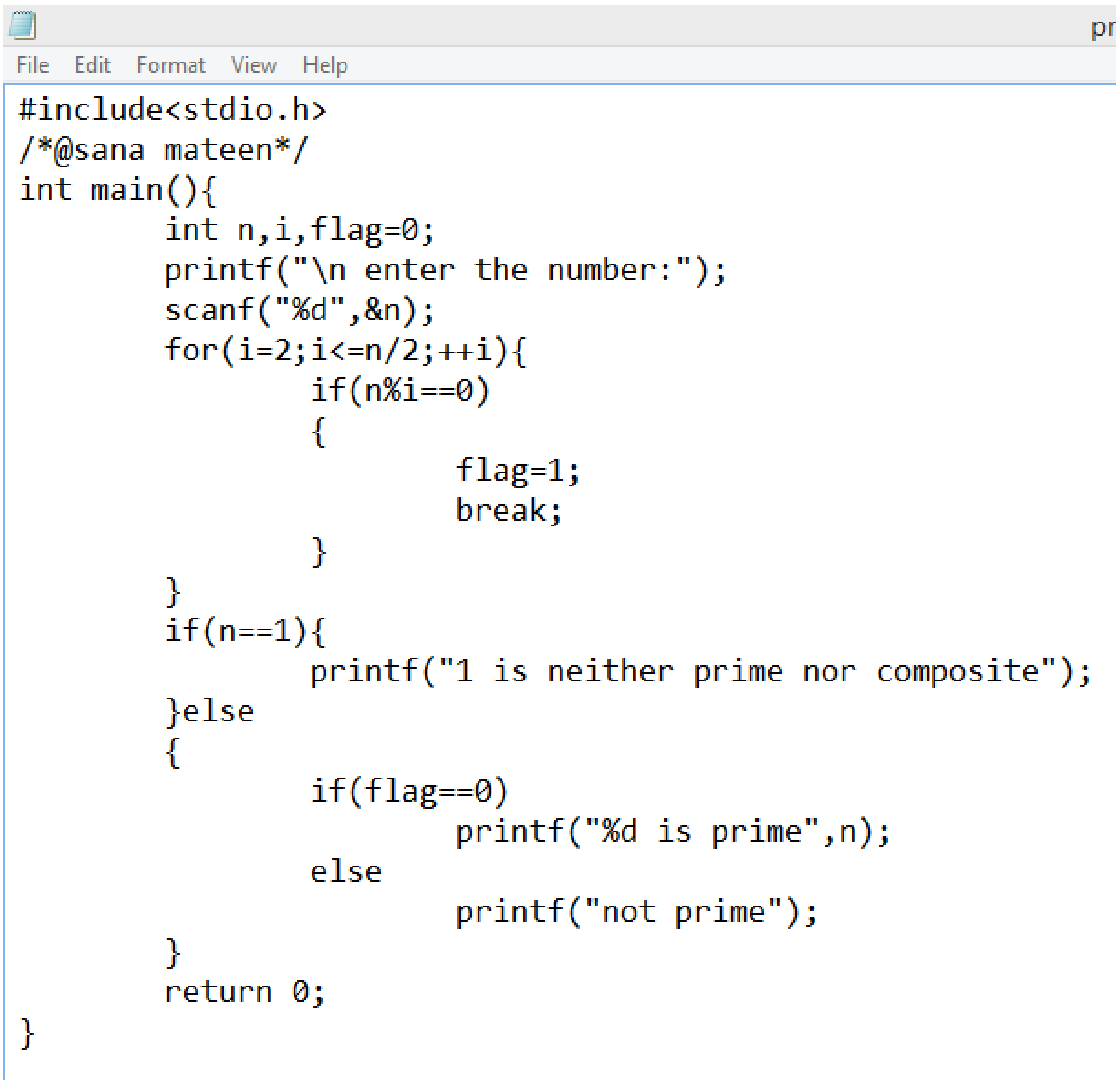
Output:



```
 enter the limit:4
0 1 1 2 3
```

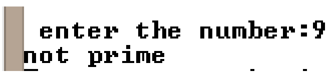
6. **Write a c program to find if a number is prime number or not?**

Program:



```
#include<stdio.h>
/*@sana mateen*/
int main(){
        int n,i,flag=0;
        printf("\n enter the number:");
        scanf("%d",&n);
        for(i=2;i<=n/2;++i){
                if(n%i==0)
                {
                        flag=1;
                        break;
                }
        }
        if(n==1){
                printf("1 is neither prime nor composite");
        }else
        {
                if(flag==0)
                        printf("%d is prime",n);
                else
                        printf("not prime");
        }
        return 0;
}
```

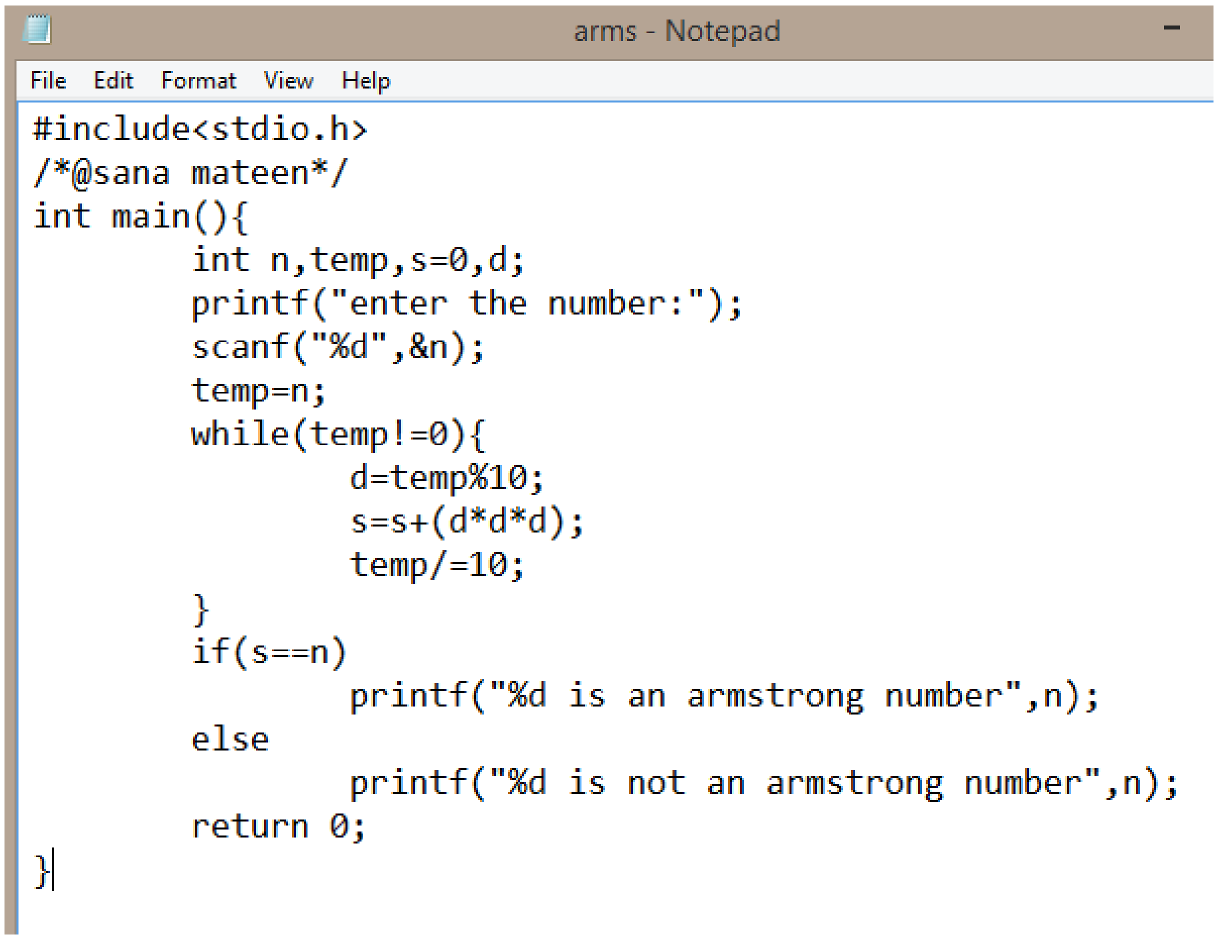
Output:



```
 enter the number:9
not prime
```

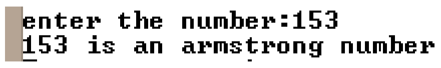
7. Write a c program to find Armstrong number?

Program:



```
#include<stdio.h>
/*@sana mateen*/
int main(){
        int n,temp,s=0,d;
        printf("enter the number:");
        scanf("%d",&n);
        temp=n;
        while(temp!=0){
                d=temp%10;
                s=s+(d*d*d);
                temp/=10;
        }
        if(s==n)
                printf("%d is an armstrong number",n);
        else
                printf("%d is not an armstrong number",n);
        return 0;
}
```

Output:



```
enter the number:153
153 is an armstrong number
```

8. What is recursion?What are its advantages?

The process in which a function calls itself directly or indirectly is called recursion.

Advantages:

- Recursion makes program elegant and cleaner.
- Better understanding of algorithm.

– Less code and small programs.

9. What is dynamic memory allocation and mention the need for it?

An array is a collection of a fixed number of values. **Once the size of an array is declared, you cannot change it.**

Sometimes the size of the array you declared may be insufficient. **To solve this issue, you can allocate memory manually during run-time.** This is known as dynamic memory allocation in C programming.

To allocate memory dynamically, library functions are malloc(), calloc(), realloc() and free() are used.

These functions **are defined in the <stdlib.h>** header file.

10. Write the syntax of the following
i)malloc ii)calloc iii)realloc iv)free

i) Syntax of malloc()

void* malloc(size_t size);

ptr = (castType*) malloc(size);

ii) Syntax of calloc()

void* calloc(size_t num, size_t size);

ptr = (castType*)calloc(n, size);

iii) Syntax of realloc()

ptr = realloc(ptr, x);

iv) Syntax of free()

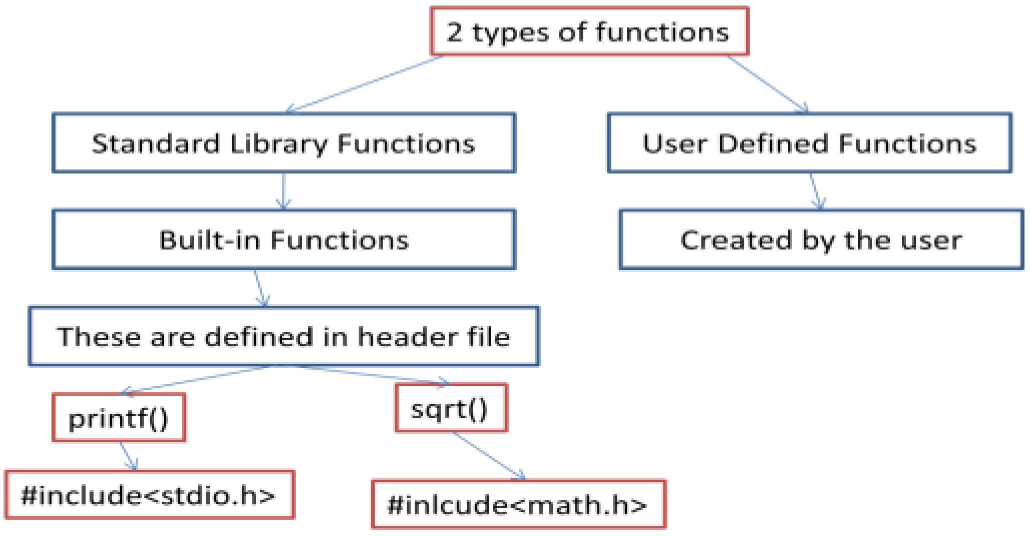
free(ptr);

Long Answer Questions for 5 Marks:

1. **Explain functions in detail along with an example program?**

A function is a self contained sub program that carries out some specific task

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



Example Program

```
•   #include<stdio.h>
•   returntype  func_name(arg_list);
•   int main(){
          ____________________

          ____________________

          func_name();
          ____________________
          ____________________
      }
      returntype   func_name(arg_list){
      //statements
      }
```

Execution of c program begins from main()

when compiler encounters func_name(), control of the program jumps to

returntype func_name();

void func_name(){

//statements

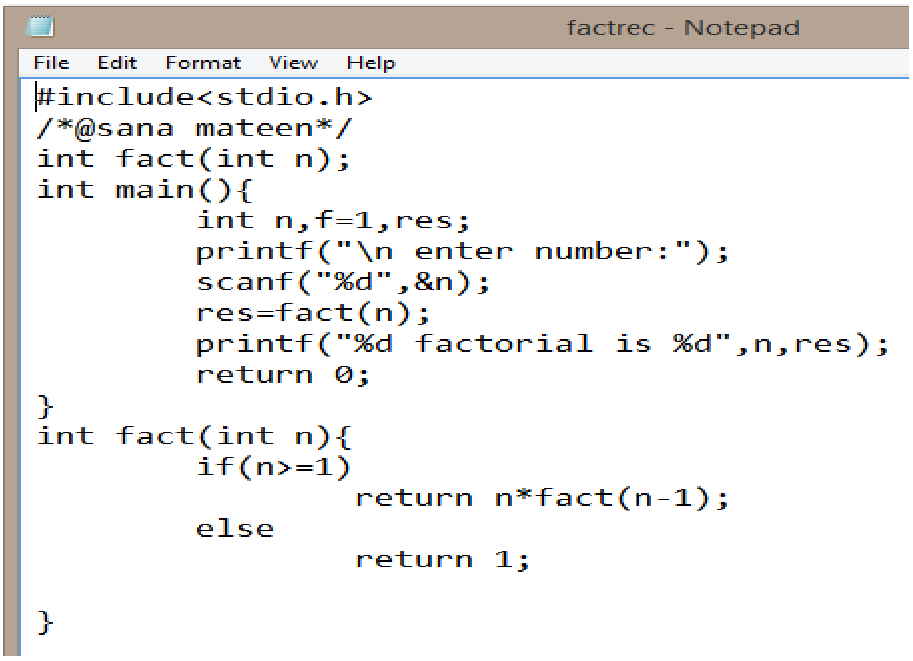
}

Advantages of userdefined function:

program will be easy to understand, maintain and debug

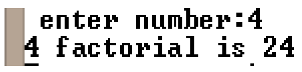
reusable code at can be used in other programs.

Program:



```
#include<stdio.h>
/*@sana mateen*/
int fact(int n);
int main(){
        int n,f=1,res;
        printf("\n enter number:");
        scanf("%d",&n);
        res=fact(n);
        printf("%d factorial is %d",n,res);
        return 0;
}
int fact(int n){
        if(n>=1)
                return n*fact(n-1);
        else
                return 1;

}
```

Output:



```
 enter number:4
4 factorial is 24
```

2. Explain actual arguments and formal arguments with the help of program?

Basically, there are two types of arguments:

Actual arguments

Formal arguments

The variables declared in the function prototype or definition are known as Formal arguments and the values that are passed to the called function from the main function are known as Actual arguments.

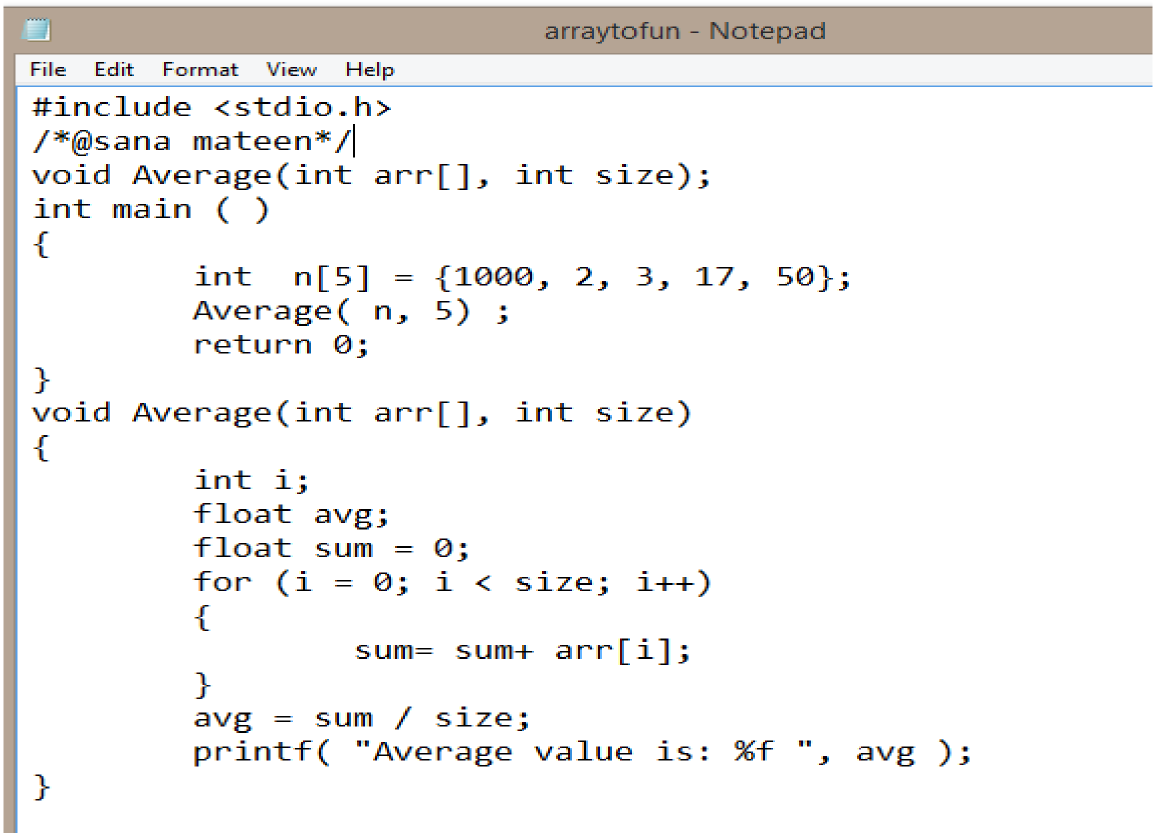
The actual arguments and formal arguments must match in number, type, and order.

Following are the two ways to pass arguments to the function:

Pass by value

Pass by reference

Program:



```
#include <stdio.h>
/*@sana mateen*/
void Average(int arr[], int size);
int main ( )
{
        int  n[5] = {1000, 2, 3, 17, 50};
        Average( n, 5) ;
        return 0;
}
void Average(int arr[], int size)
{
        int i;
        float avg;
        float sum = 0;
        for (i = 0; i < size; i++)
        {
                sum= sum+ arr[i];
        }
        avg = sum / size;
        printf( "Average value is: %f ", avg );
}
```

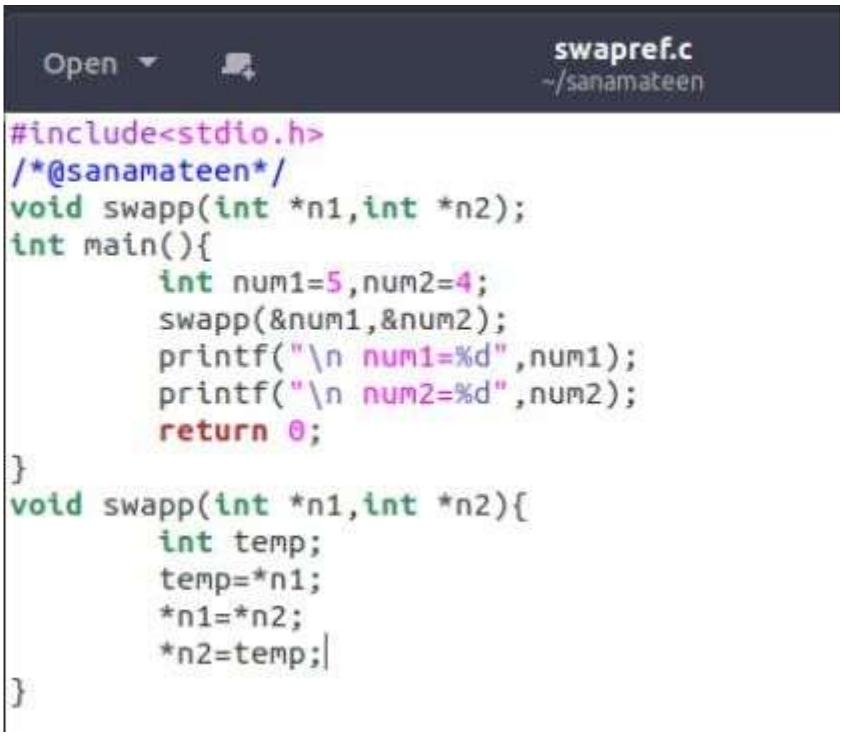
Output:

```
Average value is: 214.399994
```

3. Write a c program to swap two numbers using call by reference?

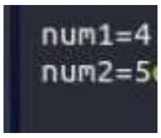
The call by reference method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. It means the changes made to the parameter affect the passed argument.

Program:



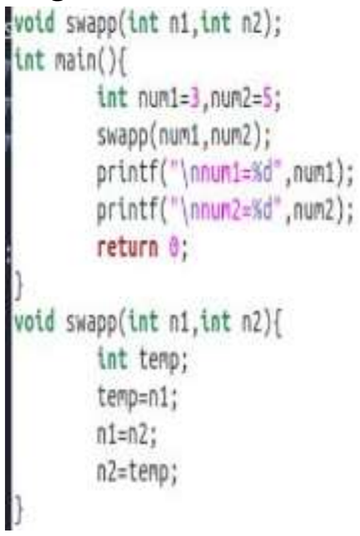
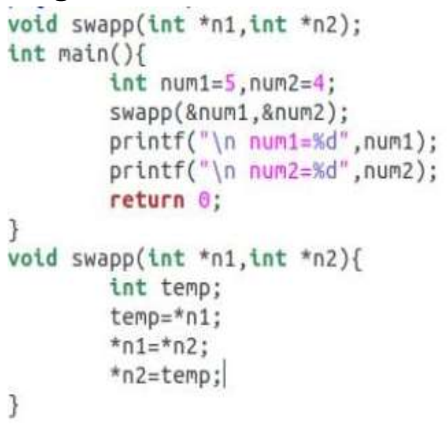
```c
#include<stdio.h>
/*@sanamateen*/
void swapp(int *n1,int *n2);
int main(){
        int num1=5,num2=4;
        swapp(&num1,&num2);
        printf("\n num1=%d",num1);
        printf("\n num2=%d",num2);
        return 0;
}
void swapp(int *n1,int *n2){
        int temp;
        temp=*n1;
        *n1=*n2;
        *n2=temp;
}
```

Output:



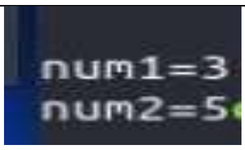
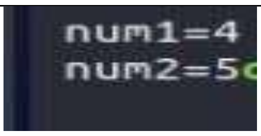
```
num1=4
num2=5
```

4. Differentiate between Call By Value and Call By Reference in detail?

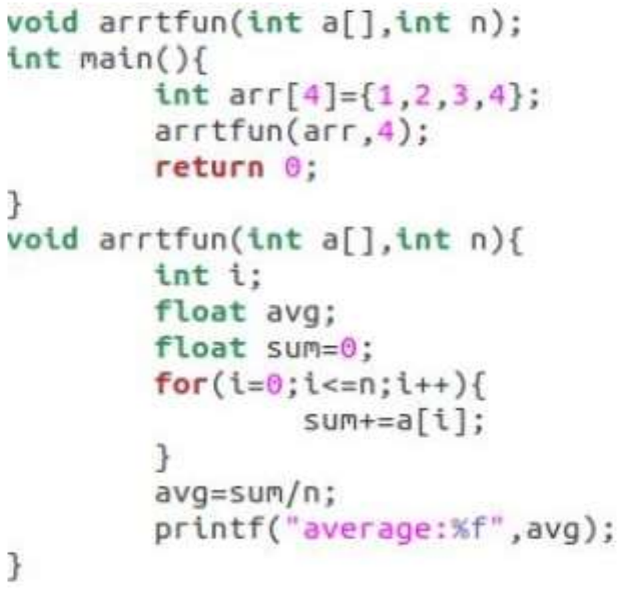
Call By Value	**Call By Reference**
In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.	In call by reference, the address of the variable is passed into the function call as the actual parameter.
In call by value method, we can not modify the value of the actual parameter by the formal parameter.	The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.	In call by reference, the memory allocation is similar for both formal parameters and actual parameters.
The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.	All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.
Program: 	**Program:** 
Output:	**Output:**

```
void swapp(int n1,int n2);
int main(){
        int num1=3,num2=5;
        swapp(num1,num2);
        printf("\nnum1=%d",num1);
        printf("\nnum2=%d",num2);
        return 0;
}
void swapp(int n1,int n2){
        int temp;
        temp=n1;
        n1=n2;
        n2=temp;
}
```

```
void swapp(int *n1,int *n2);
int main(){
        int num1=5,num2=4;
        swapp(&num1,&num2);
        printf("\n num1=%d",num1);
        printf("\n num2=%d",num2);
        return 0;
}
void swapp(int *n1,int *n2){
        int temp;
        temp=*n1;
        *n1=*n2;
        *n2=temp;
}
```

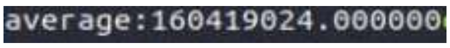
 ```num1=3 num2=5```	 ```num1=4 num2=5```

5. **Write a c program to pass arrays to a function?**
   Program:



```
void arrtfun(int a[],int n);
int main(){
        int arr[4]={1,2,3,4};
        arrtfun(arr,4);
        return 0;
}
void arrtfun(int a[],int n){
        int i;
        float avg;
        float sum=0;
        for(i=0;i<=n;i++){
                sum+=a[i];
        }
        avg=sum/n;
        printf("average:%f",avg);
}
```

   Output:



```
average:160419024.000000
```

6. **What is recursion?Explain different types of recursion along with their advantages and disadavantages?**
   The process in which a function calls itself directly or indirectly is called recursion.

Types of recursion:

1. Direct
2. Indirect

1. Direct Recursion

A function is said to be direct recursive if it calls itself directly.

```
void recursion()
{
recursion();
/* function calls itself */
}
int main()
{
 recursion();
 }
```

2. Indirect Recursion

A function is said to be indirect recursive if it calls another function and this new function calls the first calling function again.

Example:

```
int func1(int n)
{
        if (n<=1)
        return  1;
        else
        return func2(n);
}
 int func2(int n)
{           return func1(n);
}
```

In this program, func1() calls func2(), which is a new function. But this new function func2() calls the first calling function, func1(), again. This makes the above function an indirect recursive function.

Advantages:

Recursion makes program elegant and cleaner.

Better understanding of algorithm.

Less code and small programs.

Disadvantages:

Slow speed.

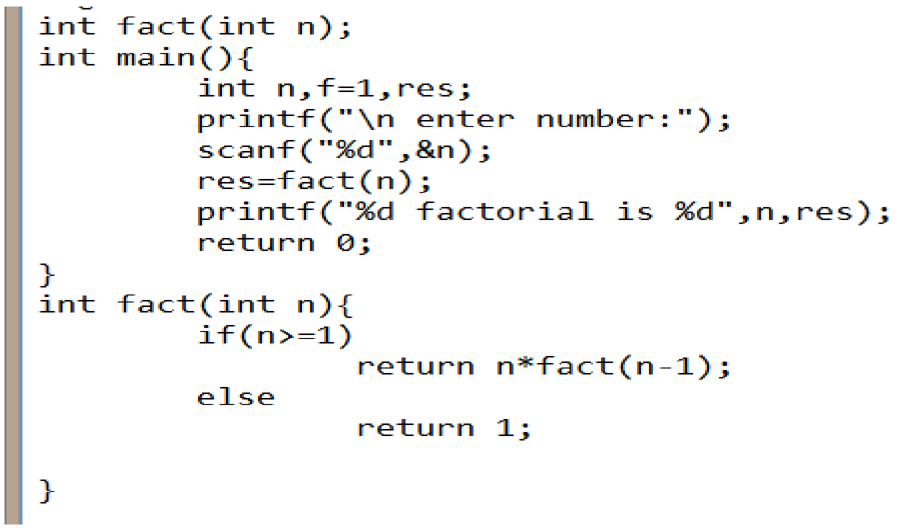
More memory space is required to store intermediate results.

7. Write a c program to find the factorial of a number using recursion?

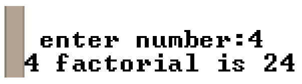
The process in which a function calls itself directly or indirectly is called recursion.

Program:

Output:



```c
int fact(int n);
int main(){
        int n,f=1,res;
        printf("\n enter number:");
        scanf("%d",&n);
        res=fact(n);
        printf("%d factorial is %d",n,res);
        return 0;
}
int fact(int n){
        if(n>=1)
                return n*fact(n-1);
        else
                return 1;

}
```



```
 enter number:4
4 factorial is 24
```

8. Explain dynamic memory allocation in detail along with an example program?

Dynamic Memory Allocation is manual allocation and freeing of memory according to your programming needs. Dynamic memory is managed and served with pointers that point to the newly allocated memory space in an area which we call the heap.

Now you can create and destroy an array of elements dynamically at runtime without any problems. To sum up, the automatic memory management uses the stack, and the dynamic memory allocation uses the heap.

The <stdlib.h> library has functions responsible for dynamic memory management.

Function	Purpose
malloc	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc	It is used to modify the size of previously allocated memory space.
Free	Frees or empties the previously allocated memory space.

The malloc Function

The malloc() function stands for memory allocation. It is a function which is used to allocate a block of memory dynamically. It reserves memory space of specified size and returns the null pointer pointing to the memory location. The pointer returned is usually of type void. It means that we can assign malloc function to any pointer.

Syntax

ptr = (cast_type *) malloc (byte_size);

Here,

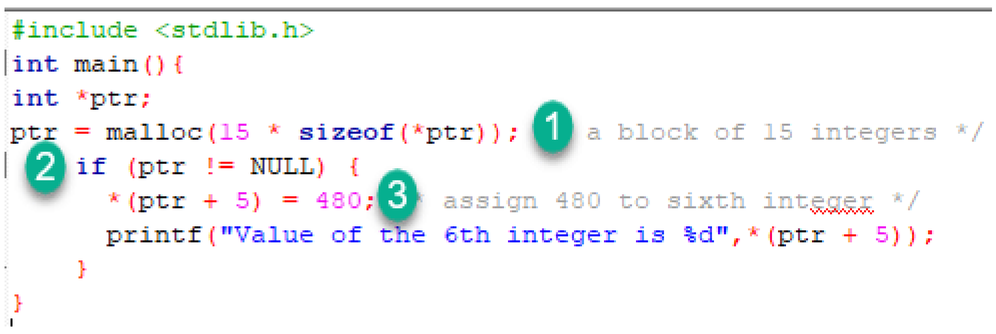
ptr is a pointer of cast_type.

The malloc function returns a pointer to the allocated memory of byte_size.

Example: ptr = (int *) malloc (50)

When this statement is successfully executed, a memory space of 50 bytes is reserved. The address of the first byte of reserved space is assigned to the pointer ptr of type int.

Consider another example:



```
#include <stdlib.h>
int main(){
int *ptr;
ptr = malloc(15 * sizeof(*ptr));    a block of 15 integers */
    if (ptr != NULL) {
      *(ptr + 5) = 480;  * assign 480 to sixth integer */
      printf("Value of the 6th integer is %d",*(ptr + 5));
    }
}
```

Notice that sizeof(*ptr) was used instead of sizeof(int) in order to make the code more robust when *ptr declaration is typecasted to a different data type later.

The allocation may fail if the memory is not sufficient. In this case, it returns a NULL pointer. So, you should include code to check for a NULL pointer.

Keep in mind that the allocated memory is contiguous and it can be treated as an array. We can use pointer arithmetic to access the array elements rather than using brackets [ ]. We advise to use + to refer to array elements because using incrementation ++ or += changes the address stored by the pointer.

Malloc function can also be used with the character data type as well as complex data types such as structures.

The free Function

The memory for variables is automatically deallocated at compile time. In dynamic memory allocation, you have to deallocate memory explicitly. If not done, you may encounter out of memory error.

The free() function is called to release/deallocate memory. By freeing memory in your program, you make more available for use later.

For example:

```
#include <stdio.h>
int main() {
int* ptr = malloc(10 * sizeof(*ptr));
if (ptr != NULL){
  *(ptr + 2) = 50;
  printf("Value of the 2nd integer is %d",*(ptr + 2));
}
free(ptr);
}
```

```
Value of the 2nd integer is 50
```

The calloc Function

The calloc function stands for contiguous allocation. This function is used to allocate multiple blocks of memory. It is a dynamic memory allocation function which is used to allocate the memory to complex data structures such as arrays and structures.

Malloc function is used to allocate a single block of memory space while the calloc function is used to allocate multiple blocks of memory space. Each block allocated by the calloc function is of the same size.

Syntax:

ptr = (cast_type *) calloc (n, size);

The above statement is used to allocate n memory blocks of the same size.

After the memory space is allocated, then all the bytes are initialized to zero.

The pointer which is currently at the first byte of the allocated memory space is returned.

Whenever there is an error allocating memory space such as the shortage of memory, then a null pointer is returned.

The program below calculates the sum of an arithmetic sequence

```
#include <stdio.h>
    int main() {
        int i, * ptr, sum = 0;
        ptr = calloc(10, sizeof(int));
        if (ptr == NULL) {
            printf("Error! memory not allocated.");
            exit(0);
        }
        printf("Building and calculating the sequence sum of the first 10 te
rms \ n ");
        for (i = 0; i < 10; ++i) { * (ptr + i) = i;
            sum += * (ptr + i);
        }
        printf("Sum = %d", sum);
        free(ptr);
        return 0;
    }
```

```
Building and calculating the sequence sum of the first 10 terms
Sum = 45
```

UNIT-5

Short Answer Questions for 2 Marks

1. **What is the need for the algorithm?**
    - To understand the basic idea of the problem.
    - To find an approach to solve the problem.
    - To improve the efficiency of existing techniques.
    - To understand the basic principles of designing the algorithms.
    - To compare the performance of the algorithm with respect to other techniques.
    - It is the best method of description without describing the implementation detail.
    - The Algorithm gives a clear description of requirements and goal of the problem to the designer.
    - A good design can produce a good solution.
    - To understand the flow of the problem.
2. **Write an algorithm to find roots of quadratic equation?**

```
step 1: Start
step 2: Read a,b,c
step 3: intilize d<-(b*b)-(4*a*c)
step 4: intilize r<-b/2*a
step 5: if d>0 go to step 6
else go to step8
step 6: r1=r+(sqrt(d)/2*a) and r2=r-(sqrt(d)/2*a)
step 7: print roots are real and distinct ,first root r1,second root r2
step 8: if d=0 go to step9
else go to step10
step 9: print roots are real and equal,-r
step 10: d=-d
step 11: im=sqrt(d)/2*a
step 12: print roots are imaginary ,first root r+i im,second root r-i im
step 13: stop
```

3. **Write an algorithm to find prime number?**

```
Step 1: Start

Step 2: Read number n

Step 3: Set f=0
Step 4: For i=2 to n-1

Step 5: If n mod 1=0 then

Step 6: Set f=1 and break

Step 7: Loop

Step 8: If f=0 then

         print 'The given number is prime'

        else

         print 'The given number is not prime'

Step 9: Stop
```

4. **What is searching technique?Mention its type along with advantages?**

   The process of identifying or finding a particular record is called Searching. You often spend time in searching for any desired item. If the data is kept properly in sorted order, then searching becomes very easy and efficient.
   Searching is an operation or a technique that helps finds the place of a given element or value in the list. Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not. Some of the standard searching technique that is being followed in the data structure is listed below:
   - Linear Search or Sequential Search
   - Binary Search

5. **Write about linear search technique and steps followed for it?**

   A linear or sequential search of an array begins at the beginning of the array and continues until the item is found or the entire array has been searched.

   - Following are the steps of implementation that we will be following:
   - Traverse the array using a for loop.
   - In every iteration, compare the target value with the current value of the array.
   - If the values match, return the current index of the array.
   - If the values do not match, move on to the next array element.
   - If no match is found, return -1.
   - To search the number 5 in the array given below, linear search will go step by step in a sequential order starting from the first element in the given array.

6. **Write about Binary search technique and steps followed for it?**

The search starts at the center of a sorted array, if center is equal to target element search is successful otherwise it determines which half to continue to search on that basis.

- The algorithm starts searching with the mid element.
- mid=(first + last)/2
- If the item is equal to mid then search is successful.
- If the item is less than the mid element, it starts over searching the first half of the list.
- If the item is greater than the mid element, it starts over searching the second half of the list.
- It then continues halving the list until the item is found.
- Each iteration eliminates half of the remaining elements.

Short Answer Questions for 3 Marks:

7. **Define sorting technique?Write an algorithm to perform bubblesort?**

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats

Algorithm:

Following are the steps involved in bubble sort(for sorting a given array in ascending order):

1.Starting with the first element(index = 0), compare the current element with the next element of the array.

2.If the current element is greater than the next element of the array, swap them.

3.If the current element is less than the next element, move to the next element. Repeat Step 1.

8. **Explain selection sort algorithm ?**

Following are the steps involved in selection sort(for sorting a given array in ascending order):

- Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.
- We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.
- We replace the element at the second position in the original array, or we can say at the first position in the subarray, with the second smallest element.

- This is repeated, until the array is completely sorted.

9. Explain insertion sort algorithm?

Insertion sort algorithm picks elements one by one and places it to the right position where it belongs in the sorted list of elements.

Using insertion sort an element is inserted in correct location.

Insertion sort scans the list from A[0] to A[n-1], insert each element A[k] into its proper position in previously sorted sub list A[0],A[1],A[2]…A[k-1]..

Procedure:

- A[0] by itself is trivially sorted.
- A[1] is inserted either before or after A[0] so that A[0],A[1] is sorted.
- A[2] is inserted into its proper place in A[0],A[1], that is, before A[0], between A[0] and A[1], or after A[1] so that A[0],A[1],A[2] is sorted.
- Repeatedly compare A[k] to the element just to the left of the vacancy, and as long as A[k] is smaller, move that element into the vacancy, else put A[k] in the vacancy.
- Repeat the next element that has not yet examined.

10. Define Time Complexity? Define Space Complexity?

Time Complexity:

The time complexity is the number of operations an algorithm performs to complete its task with respect to input size (considering that each operation takes the same amount of time).

• The algorithm that performs the task in the smallest number of operations is considered the most efficient one.

Space Complexity:

Space Complexity of an algorithm denotes the total space used or needed by the algorithm for its working, for various input sizes.

11. What is quick sort algorithm?

QuickSort is one of the most efficient sorting algorithms and is based on the splitting of an array into smaller ones. The name comes from the fact that, quick sort is capable of sorting a list of data elements significantly faster than any of the common sorting algorithms. And like Merge sort, Quick sort also falls into the category of divide and conquer approach of problem-solving methodology.

12. What is merge sort algorithm?

The MergeSort function repeatedly divides the array into two halves until we reach a stage where we try to perform MergeSort on a subarray of size 1 i.e. p == r.

After that, the merge function comes into play and combines the sorted arrays into larger arrays until the whole array is merged.

Long Answer Questions for 5 Marks:

1. Write an algorithm to perform arithmetic operation?

```
1.  Start
2.  Define a,b,c,d
3.  Give choices Addition,Subtraction,Multiplication,Division
4.  Enter choice
5.  Enter Two Numbers
6.  Check for choice
7.  If choice==1
         printf num1+num2
         step 12
       else step 8
8. If choice==2
         printf num1-num2
         step 12
       else step 9
9. If choice ==3
         printf num1*num2
         step 12
       else step 10
10. If choice==4
         printf num1/num2
         step 12
       else step 11
```

11. Choice is invalid

12. Stop

2. Write an algorithm to find the greatest among three numbers?

```
Step 1: Start
Step 2: Declare variables a,b and c.
Step 3: Read variables a,b and c.
Step 4: If a > b
            If a > c
              Display a is the largest number.
            Else
              Display c is the largest number.
          Else
            If b > c
              Display b is the largest number.
            Else
              Display c is the greatest number.
Step 5: Stop
```

3. Write an algorithm to perform binary search?

Binary search algorithm works on the principle of divide and conquer. This algorithm works with the sorted data. Binary search looks for an element to be searched in the middle first, if the element matches with the required one then it is returned as an output otherwise the whole list is considered into two parts i.e left side list and right side list.

If the element to be searched is greater than an element in the middle position then required element is searched into left side of the array otherwise element is searched on the left side of the array. This process is carried on until element from an array matches the required element.

```
Step 1: [INITIALIZE] SET BEG = lower_bound
          END = upper_bound, POS = - 1
Step 2: Repeat Steps 3 and 4 while BEG <= END
Step 3: SET MID = (BEG + END)/2
Step 4: IF A[MID] = VAL
                  SET POS = MID
                  PRINT POS
                  Go to Step 6
          ELSE IF A[MID] > VAL
```

```
                SET END = MID - 1
        ELSE
                SET BEG = MID + 1
        [END OF IF]
[END OF LOOP]
Step 5: IF POS = -1
        PRINT "VALUE IS NOT PRESENT IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

4. **Write an algorithm to perform linear search?**

Linear search is a very basic and simple search algorithm. In Linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found.

As we learned in the previous tutorial that the time complexity of Linear search algorithm is O(n), we will analyse the same and see why it is O(n) after implementing it.
Implementing Linear Search
Following are the steps of implementation that we will be following:

Traverse the array using a for loop.
In every iteration, compare the target value with the current value of the array.
If the values match, return the current index of the array.
If the values do not match, move on to the next array element.
If no match is found, return -1.
To search the number 5 in the array given below, linear search will go step by step in a sequential order starting from the first element in the given array.

```
Linear Search ( Array A, Value x)

Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit
```

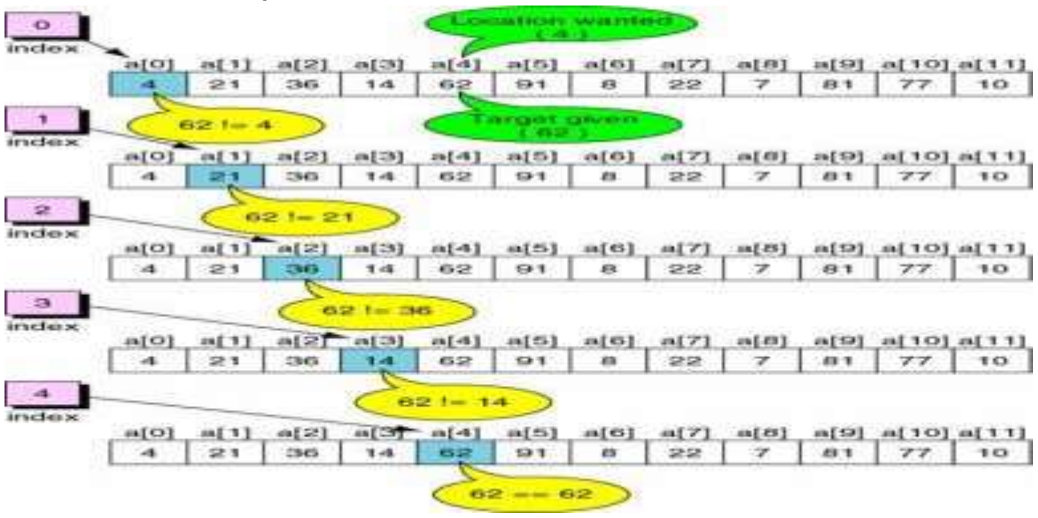
5. **Write an algorithm to find roots of quadratic equations?**

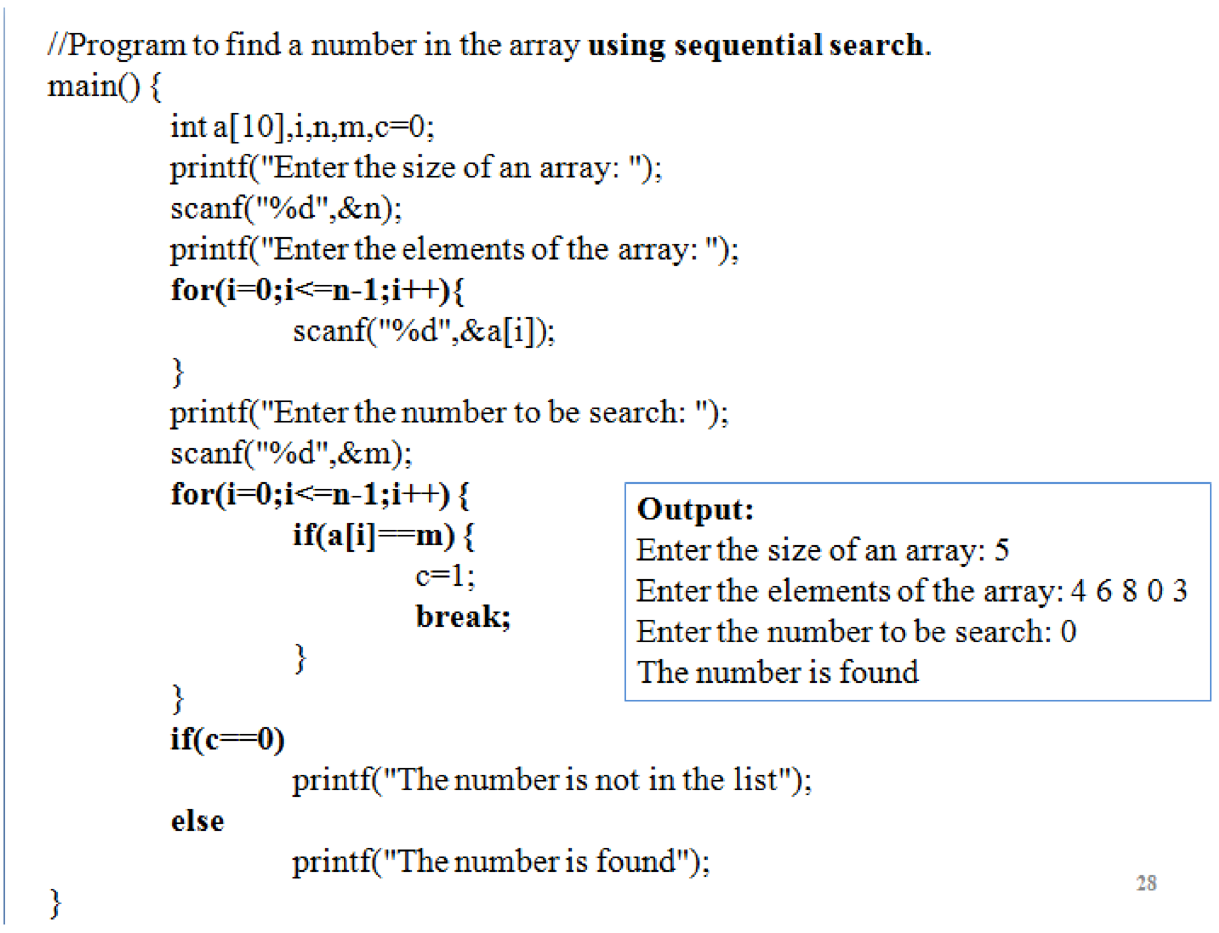
```
Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

    D ← b2-4ac

Step 4: If D ≥ 0

        r1 ← (-b+√D)/2a

        r2 ← (-b-√D)/2a

        Display r1 and r2 as roots.

    Else

        Calculate real part and imaginary part

        rp ← -b/2a

        ip ← √(-D)/2a

        Display rp+j(ip) and rp-j(ip) as roots

Step 5: Stop
```

6. Explain linear search along with an example program?

A **linear or sequential search** of an array begins at the beginning of the array and continues until the item is found or the entire array has been searched.

- ➢ It is the basic searching technique.
- ➢ Very easy to implement
- ➢ The array DOESN'T have to be sorted.
- ➢ All array elements must be visited if the search fails.
- ➢ Could be very slow.





```
//Program to find a number in the array using sequential search.
main() {
        int a[10],i,n,m,c=0;
        printf("Enter the size of an array: ");
        scanf("%d",&n);
        printf("Enter the elements of the array: ");
        for(i=0;i<=n-1;i++){
                scanf("%d",&a[i]);
        }
        printf("Enter the number to be search: ");
        scanf("%d",&m);
        for(i=0;i<=n-1;i++) {
                if(a[i]==m) {
                        c=1;
                        break;
                }
        }
        if(c==0)
                printf("The number is not in the list");
        else
                printf("The number is found");
}
```

Output:

```
Enter the size of an array: 5
Enter the elements of the array: 4 6 8 0 3
Enter the number to be search: 0
The number is found
```

7. Explain binary search along with an example program?

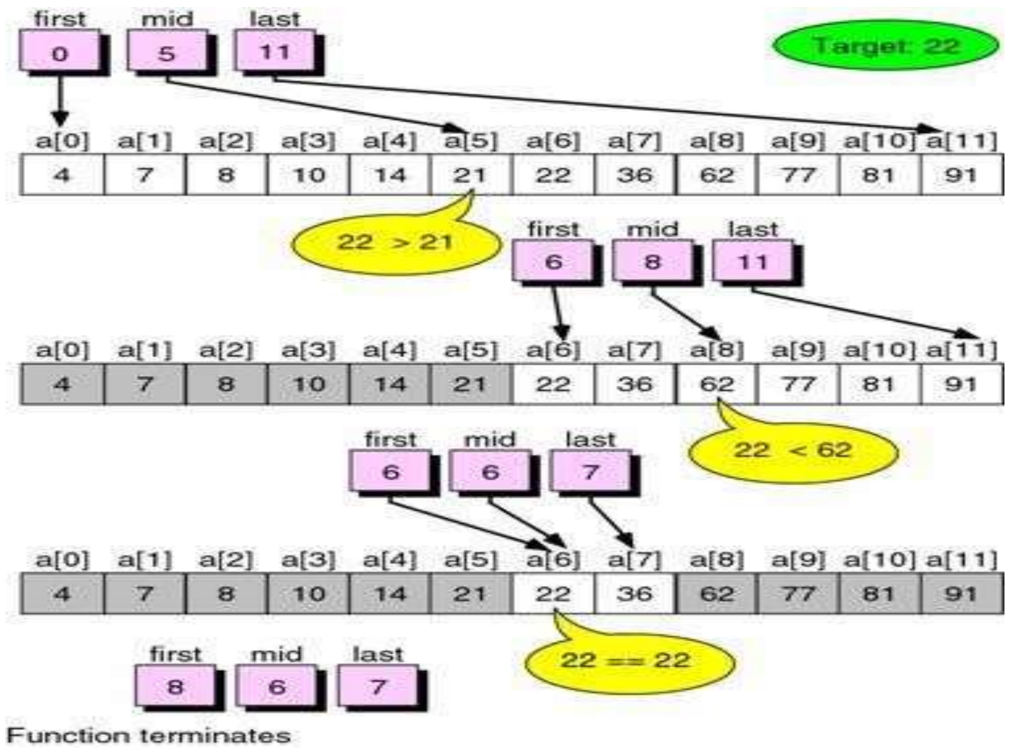
The search starts at the center of a sorted array, if center is equal to target element search is successful otherwise it determines which half to continue to search on that basis.

- The algorithm starts searching with the mid element.

mid=(first + last)/2

  - If the item is equal to mid then search is successful.
  - If the item is less than the mid element, it starts over searching the first half of the list.
  - If the item is greater than the mid element, it starts over searching the second half of the list.
  - It then continues halving the list until the item is found.
- Each iteration eliminates half of the remaining elements.
- It is faster than the linear search.
- It works only on SORTED array.

- Thus, there is a performance penalty for sorting the array.



```
/* Program to find a number in the array using binary search
#include <stdio.h>
void main() {
int list[8], target, index= -1,i,SIZE=8,low=0,high,mid;
printf("Enter %d elements in ascending or descending order: ",SIZE);
for(i=0;i<SIZE;i++)
        scanf("%d",&list[i]);
printf("Enter an element that is to be searched: ");
scanf("%d", &target);
high=SIZE-1;
while ( high >= low )  {
        mid = ( low + high ) / 2;
        if ( target < list[mid] ){  high = mid - 1;}
        else
        if ( target > list[mid] ) {low = mid + 1;}
        else            {
                index=mid;      break;
        }
}
if (index != -1)
        printf("\nTarget was found at index: %d ", index);
else
        printf("Sorry, target item was not found");
}
```

8. What is sorting?What is the need for sorting?Mention the different types of sorting algorithm and write a c program to perform bubble sort?

Sorting:

Sorting arranges data in a sequence which makes searching easier.

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

A sorting algorithm will put items in a list into an order, such as alphabetical or numerical order. For example, a list of customer names could be sorted into alphabetical order by surname, or a list of people could be put into numerical order by age.

Need for Sorting:

Sorting a list of items can take a long time, especially if it is a large list. A computer program can be created to do this, making sorting a list of data much easier.

Types:

There are many different techniques available for sorting, differentiated by their efficiency and space requirements. Following are some sorting techniques.

- **Bubble Sort**
- **Insertion Sort**
- **Selection Sort**
- **Quick Sort**
- **Merge Sort**
- **Heap Sort**

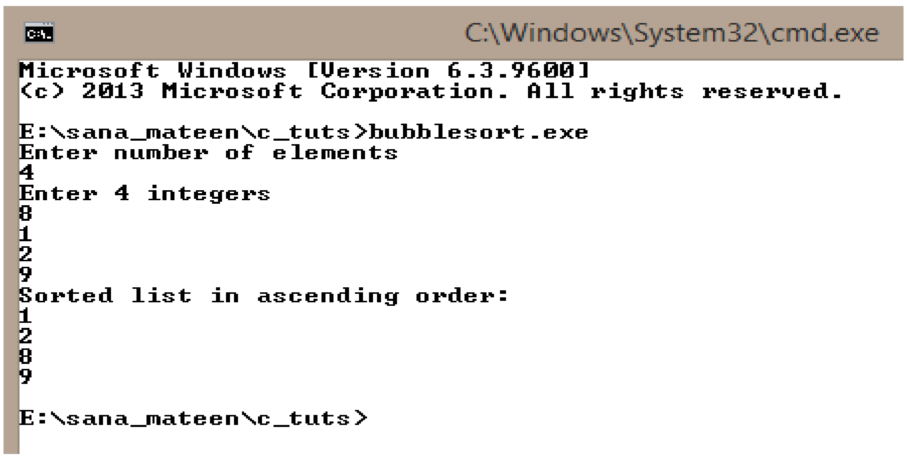
BubbleSort:

Bubble sort is an algorithm that compares the adjacent elements and swaps their positions if they are not in the intended order. The order can be ascending or descending.

Program:

```
#include<stdio.h>
/*@sana mateen*/
void main()
{
        int n, a[20], temp, i, j;
        printf("Enter the size of the array\n");
        scanf("%d", &n);
        printf("Enter the array elements\n");
        for(i = 0; i < n; i++)
        {
                scanf("%d", &a[i]);
        }
        for(i = 0; i < n - 1; i++)
        {
                for(j = 0; j < n - 1; j++)
                {
                        if(a[j] > a[j + 1])
                        {
                                temp = a[j];
                                a[j] = a[j + 1];
                                a[j + 1] = temp;
                        }
                }
        }
        printf("The sorted array is\n");
        for(i = 0; i < n; i++)
        printf("%d\n", a[i]);

}
```

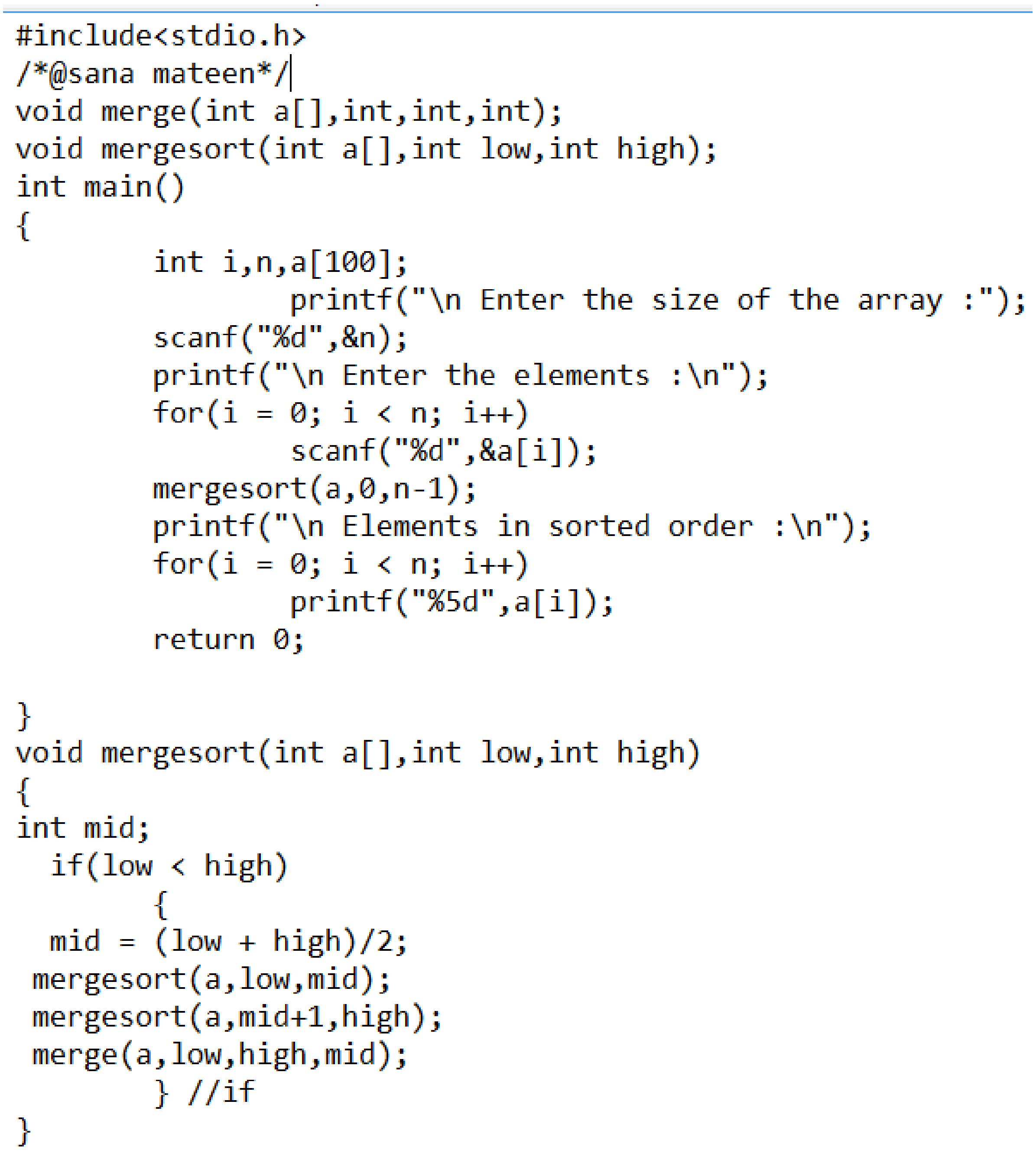
Output:



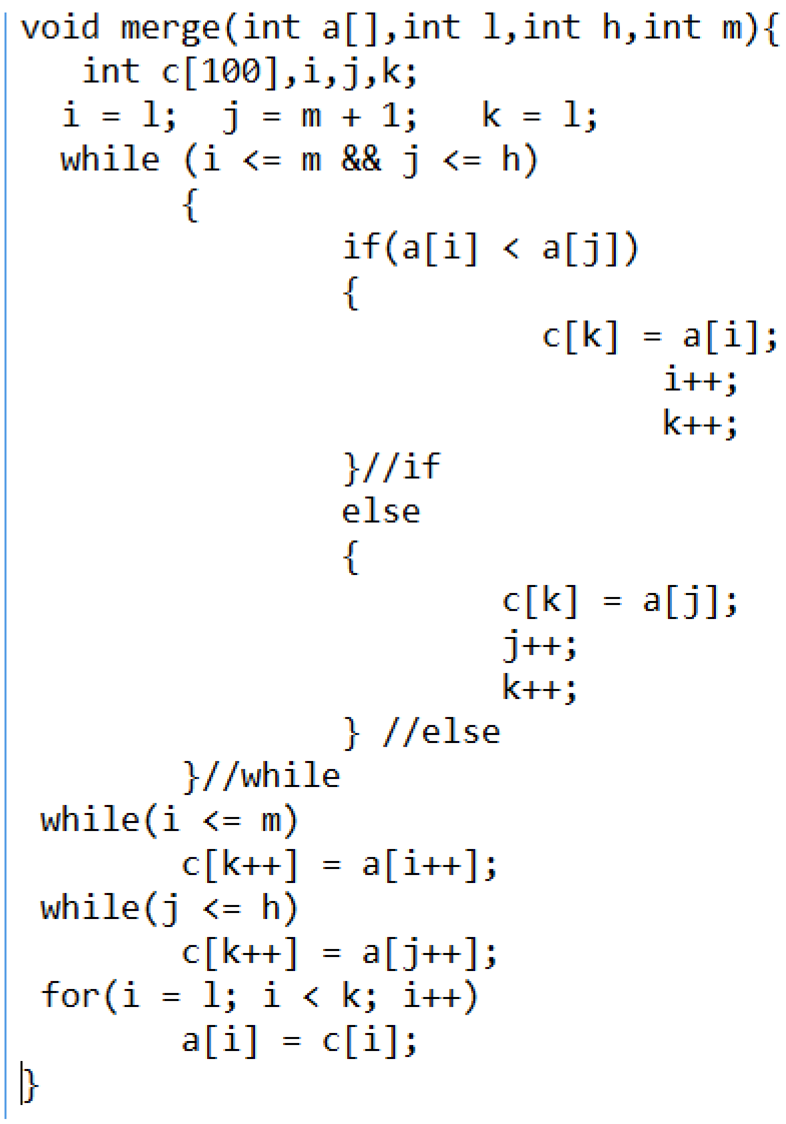
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

E:\sana_mateen\c_tuts>bubblesort.exe
Enter number of elements
4
Enter 4 integers
8
1
2
9
Sorted list in ascending order:
1
2
8
9

E:\sana_mateen\c_tuts>
```

9. Write a c program to perform merge sort?

Program:

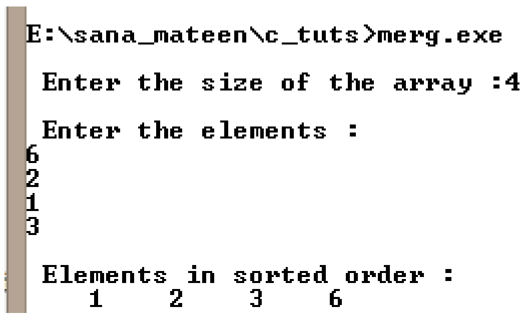


```
#include<stdio.h>
/*@sana mateen*/
void merge(int a[],int,int,int);
void mergesort(int a[],int low,int high);
int main()
{
        int i,n,a[100];
                printf("\n Enter the size of the array :");
        scanf("%d",&n);
        printf("\n Enter the elements :\n");
        for(i = 0; i < n; i++)
                scanf("%d",&a[i]);
        mergesort(a,0,n-1);
        printf("\n Elements in sorted order :\n");
        for(i = 0; i < n; i++)
                printf("%5d",a[i]);
        return 0;

}
void mergesort(int a[],int low,int high)
{
int mid;
  if(low < high)
        {
  mid = (low + high)/2;
 mergesort(a,low,mid);
 mergesort(a,mid+1,high);
 merge(a,low,high,mid);
        } //if
}
```



```
void merge(int a[],int l,int h,int m){
   int c[100],i,j,k;
  i = l;  j = m + 1;   k = l;
  while (i <= m && j <= h)
        {
                if(a[i] < a[j])
                {
                           c[k] = a[i];
                                 i++;
                                 k++;
                }//if
                else
                {
                         c[k] = a[j];
                         j++;
                         k++;
                } //else
        }//while
 while(i <= m)
        c[k++] = a[i++];
 while(j <= h)
        c[k++] = a[j++];
 for(i = l; i < k; i++)
        a[i] = c[i];
}
```

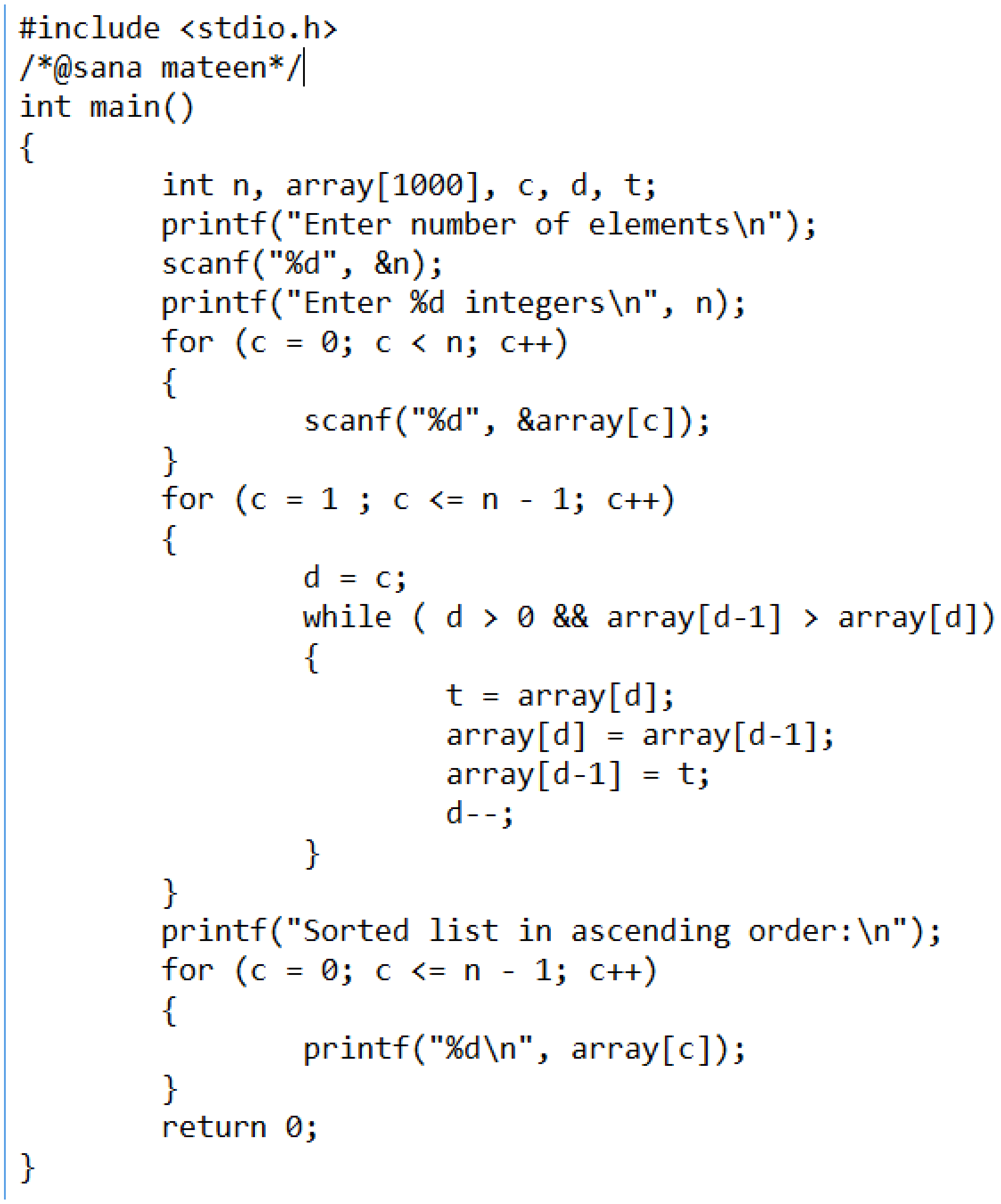
Output:



```
E:\sana_mateen\c_tuts>merg.exe
 Enter the size of the array :4
 Enter the elements :
6
2
1
3

 Elements in sorted order :
    1    2    3    6
```

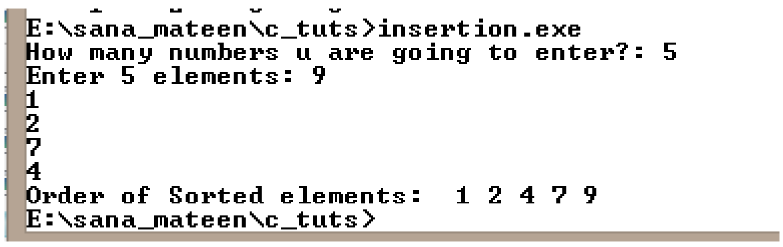
10. Write a c program to perform insertion sort?

Program:



```c
#include <stdio.h>
/*@sana mateen*/
int main()
{
        int n, array[1000], c, d, t;
        printf("Enter number of elements\n");
        scanf("%d", &n);
        printf("Enter %d integers\n", n);
        for (c = 0; c < n; c++)
        {
                scanf("%d", &array[c]);
        }
        for (c = 1 ; c <= n - 1; c++)
        {
                d = c;
                while ( d > 0 && array[d-1] > array[d])
                {
                        t = array[d];
                        array[d] = array[d-1];
                        array[d-1] = t;
                        d--;
                }
        }
        printf("Sorted list in ascending order:\n");
        for (c = 0; c <= n - 1; c++)
        {
                printf("%d\n", array[c]);
        }
        return 0;
}
```

Output:



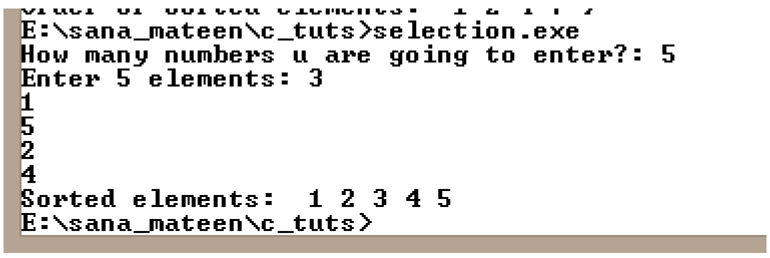
```
E:\sana_mateen\c_tuts>insertion.exe
How many numbers u are going to enter?: 5
Enter 5 elements: 9
1
2
7
4
Order of Sorted elements:  1 2 4 7 9
E:\sana_mateen\c_tuts>
```

11. Write a c program to perform selection sort?

Program:

```c
#include<stdio.h>
int main(){
   /* Here i & j for loop counters, temp for swapping,
    * count for total number of elements, number[] to
    * store the input numbers in array. You can increase
    * or decrease the size of number array as per requirement
    */
   int i, j, count, temp, number[25];

   printf("How many numbers u are going to enter?: ");
   scanf("%d",&count);
   printf("Enter %d elements: ", count);
   // Loop to get the elements stored in array
   for(i=0;i<count;i++)
      scanf("%d",&number[i]);
   // Logic of selection sort algorithm
   for(i=0;i<count;i++){
      for(j=i+1;j<count;j++){
         if(number[i]>number[j]){
            temp=number[i];
            number[i]=number[j];
            number[j]=temp;
         }
      }
   }
   printf("Sorted elements: ");
   for(i=0;i<count;i++)
      printf(" %d",number[i]);
   return 0;
}
```

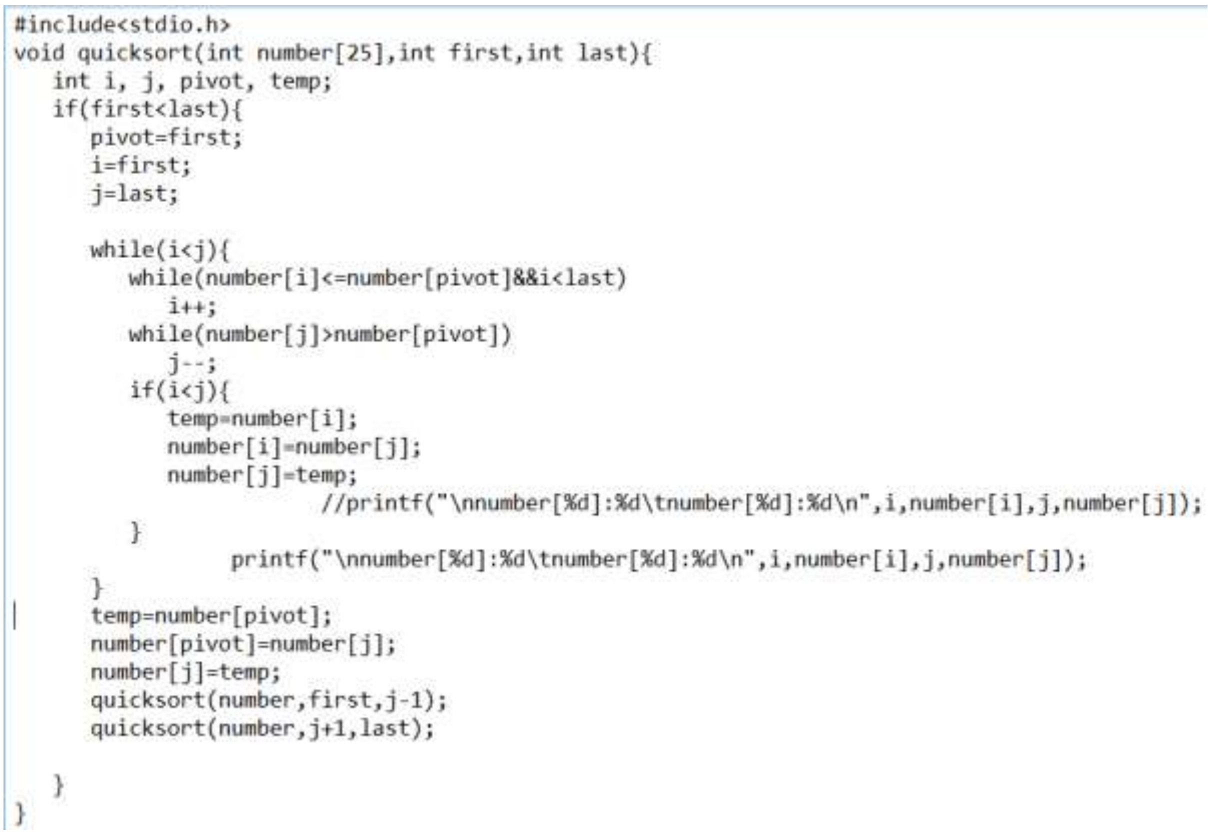
Output:



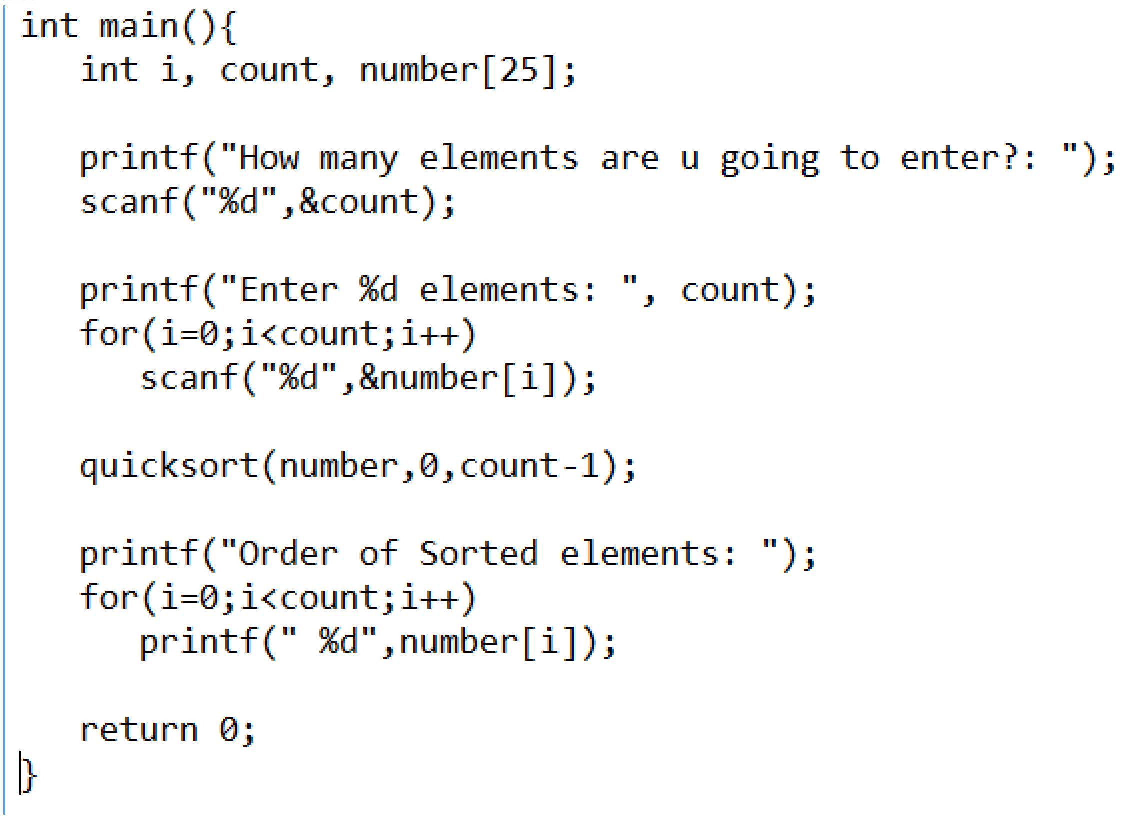
```
E:\sana_mateen\c_tuts>selection.exe
How many numbers u are going to enter?: 5
Enter 5 elements: 3
1
5
2
4
Sorted elements:  1 2 3 4 5
E:\sana_mateen\c_tuts>
```

12. Write a c program to perform quick sort?

Program:

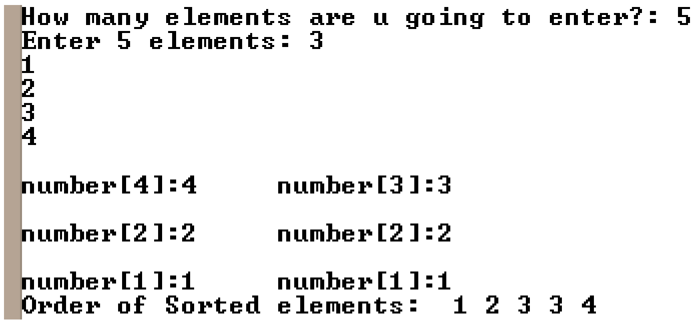


```c
#include<stdio.h>
void quicksort(int number[25],int first,int last){
   int i, j, pivot, temp;
   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(number[i]<=number[pivot]&&i<last)
            i++;
         while(number[j]>number[pivot])
            j--;
         if(i<j){
            temp=number[i];
            number[i]=number[j];
            number[j]=temp;
                        //printf("\nnumber[%d]:%d\tnumber[%d]:%d\n",i,number[i],j,number[j]);
         }
                printf("\nnumber[%d]:%d\tnumber[%d]:%d\n",i,number[i],j,number[j]);
      }
      temp=number[pivot];
      number[pivot]=number[j];
      number[j]=temp;
      quicksort(number,first,j-1);
      quicksort(number,j+1,last);

   }
}
```



```c
int main(){
   int i, count, number[25];

   printf("How many elements are u going to enter?: ");
   scanf("%d",&count);

   printf("Enter %d elements: ", count);
   for(i=0;i<count;i++)
      scanf("%d",&number[i]);

   quicksort(number,0,count-1);

   printf("Order of Sorted elements: ");
   for(i=0;i<count;i++)
      printf(" %d",number[i]);

   return 0;
}
```

Output:



```
How many elements are u going to enter?: 5
Enter 5 elements: 3
1
2
3
4

number[4]:4     number[3]:3

number[2]:2     number[2]:2

number[1]:1     number[1]:1
Order of Sorted elements:  1 2 3 3 4
```