PARUL INSTITUTE OF ENGINEERING AND TECHNOLOGY

DIPLOMA STUDIES COMPUTER

DEPARTMENT

1st SEMESTER

QUESTION BANK WITH SOLUTION

COMPUTER PROGRAMMING

(03606103)

QUESTIONS

UNIT-1- FLOW CHART & ALGORITHM

1. Definition of flowchart. (2 M)
2. Definition of an algorithm? Write an algorithm to add two number.(3 m)
4. Explain advantages & disadvantages of flowchart.(2 m)
5. Define symbols of flowchart(2 m)
6. Write an algorithm to find area of circle.(4 M)
7. Draw a flowchart to find factorial of given number. (4 M)
8. Draw a flowchart to print Fibonacci series. (4 M)
9. Draw a flowchart to find maximum out of three numbers. (4 M)
10. Draw a flowchart to display that given number is odd or even. (4 M)
11. Write an algorithm to find square of given number. (4 M)
12. Write an algorithm to find area of rectangle.(4 M)

UNIT-2- Basics of 'C'

1.Explain basic structure of C Program. (3 M)
2. Write advantages of C language. (2 M)
3. Explain Tokens of C language. (2 M)
4. What is declaration & initialization of variable? Give Example.(4 M)
5.Write the rules for defining a variable. (2 M)
6.Write a C program to print "Hello World" on your screen.(3 M)
7. Explain type conversion with example. (4 M)
8.Write a C program to find area of circle using Constant. (4 M)
9.Explain keywords of C. (2 M)
10.Write to program to check that given number is positive or negative. (4M)
11.Write a program to find given number is odd or even. (4 M)
12.What is identifiers? (2 M)
13.Explain the types of constant.(2 M)

14. Explain with an example of dynamic initialization of variables. (4 M)
15. Explain modifiers in C. (2 M)

UNIT-3 OPERATOR AND EXPRESSIONS

1. What is operator? List various operators available in c. (2 M)
2. Explain Arithmetic operators with examples. (4 M)
3. Explain Relational operators with examples. (4 M)
4. Explain Logical operators with example. (4 M)
5. Explain Assignment operators with example. (4 M)
6. Explain conditional operators with example. (4 M)
7. Explain Bitwise operators with example. (4 M)
8. Write a program to design a calculator using arithmetic operator. (4 M)
9. Write a program to swapping two numbers using Bitwise Operator. (4 M)

UNIT-4- DECISION STATEMENT

1. Explain simple If statements. (2 M)
2. Explain If….Else statements. (3 M)
3. Explain Nested if-else statement with example. (4 M)
4. Explain If else-if ladder with example. (4 M)
5. Explain Switch …case statement with example. (4 M)
6. Explain Break statement with example. (4 M)
7. What is the difference Between If…Else and Switch case. (4M)
8. Give the difference between Break ,Continue. (4 M)
9. Write a program to find given no is odd or even. (4 M)
10. Write a program to display days using switch case. (4 M)
11. Write a program to find if a given number is positive or negative. (4M)

UNIT- 5 LOOP CONTROL STATEMENTS

1. Define loop control statements and list its types. (3 M)
2. Write a c program to print sum of first 10 nos. Using for loop. (4 M)
3. State advantages of using nested for loop. (3 M)
4. Differentiate while and do while loop. (3 M)
5. Explain WHILE LOOP with Example. (4 M)
6. Explain DO…WHILE LOOP with example. (4 M)
7. Write a c program using nested for loops to find prime numbers between 2 to 100. (4 M)

ANSWER

UNIT-1- FLOW CHART & ALGORITHM

1. Definition of flowchart. (2 m)

Ans:

- Flowchart is a diagrammatic representation of an algorithm. Flowchart is very helpful in writing program and explaining program to others.
- Flow charts are drawn using certain special purpose symbols such as Rectangles, Diamonds, Ovals and small circles. These symbols are connected by arrows called flow lines.

2. Definition of an algorithm? Write an algorithm to add two number. (3 m)

Ans:

- An algorithm is a set of instructions for solving logical and mathematical problems. It is chalked out step-by-step approach to solve a given problem. It takes inputs and produces an output.
- The characteristics of a good algorithm are:
  - Precision – the steps are precisely stated (defined).
  - Uniqueness – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
  - Finiteness – the algorithm stops after a finite number of instructions are executed.
  - Input – the algorithm receives input.
  - Output – the algorithm produces output.
  - Generality – the algorithm applies to a set of inputs.

➢ **Write an algorithm to add two number.**

Step 1: Start
Step 2:Read the two numbers in to a,b
Step 3: c=a+b
Step 4: write/print c
Step 5: Stop.

3. Explain advantages & disadvantages of flowchart.(2 m)

Ans:

➢ **Advantages of Flowchart**

- It is a convenient method of communication.
- It indicates very clearly just what is being done, where a program has logical complexities.

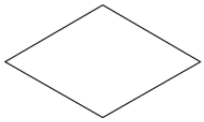
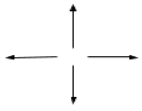
- A key to correct programming.
- It is an important tool for planning and designing a new system.
- It clearly indicates the role-played at each level.
- It saves the inconveniences in future and serves the purpose of documentation for a system.
- It promotes logical accuracy.
- It makes sure that no logical path is left incomplete without any action being taken.

➢ **Disadvantages of Flowchart**

- The flowchart is a waste of time and slows down the process of software development.
- The flowchart is quite costly to produce and difficult to use and manage.
- Flowcharts are not meant for man to computer communication.
- If you need to modify or alternate the process then it will be very hard to do in the flowchart. Because either you will have to erase the end of the flowchart or start.'

4. Define symbols of flowchart. (2 m)

Ans:

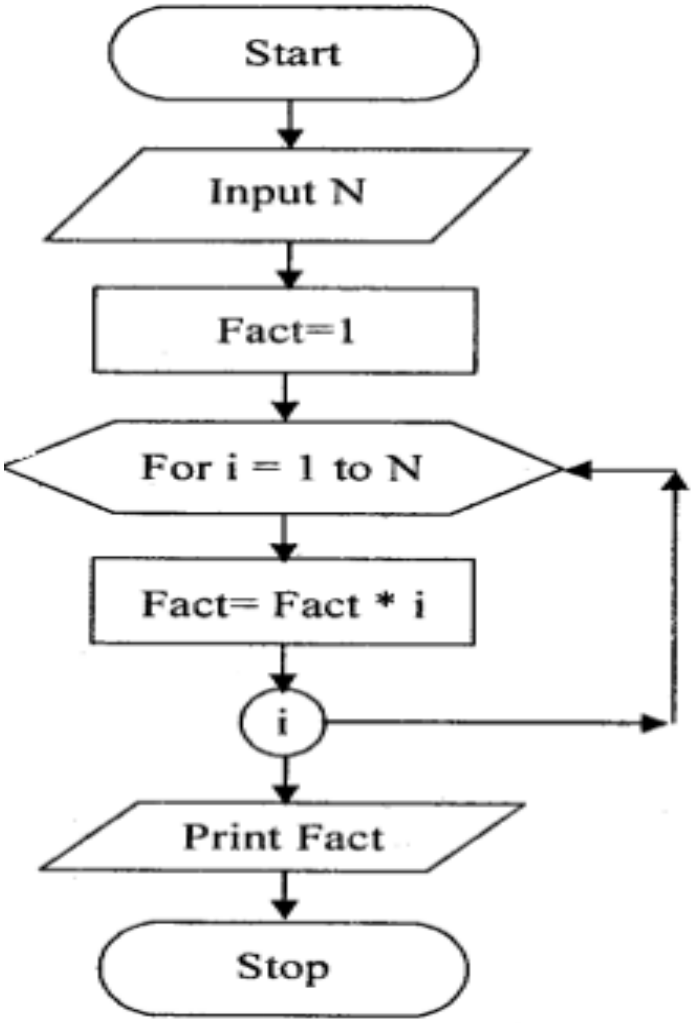
Symbol	Description
	Start / Stop
	Input / Output (Read / Print)
	Process
	Decision Making
	Subroutine
	Direction

5. Write an algorithm to find area of circle. (4 M)
Ans:

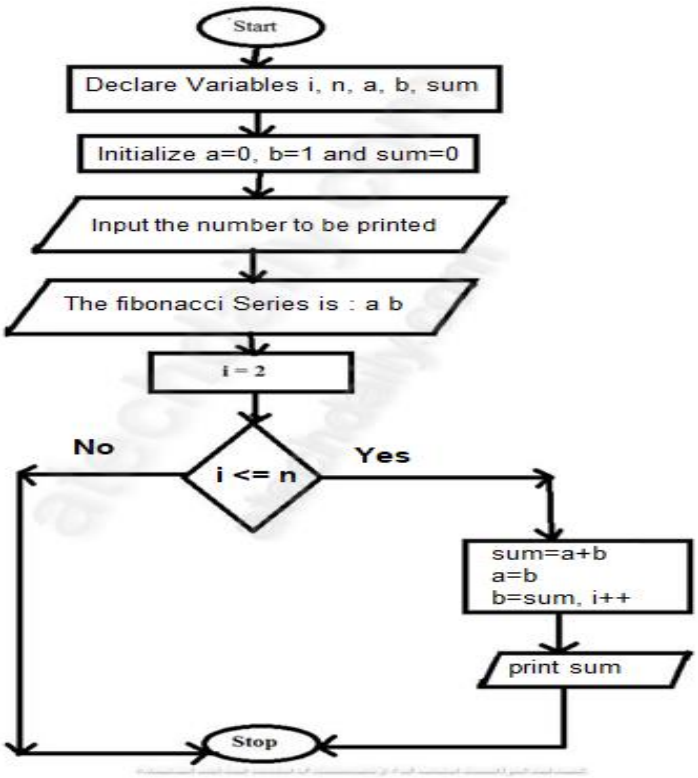
Steps of Algorithm:

1. START.
2. INTEGER AREA,RADIUS.
3. PRINT "ENTER THE RADIUS OF CIRCLE - "
4. AREA=3.14*RADIUS*RADIUS.
5. PRINT "AREA OF CIRCLE = "
6. PRINT AREA.
7. EXIT.

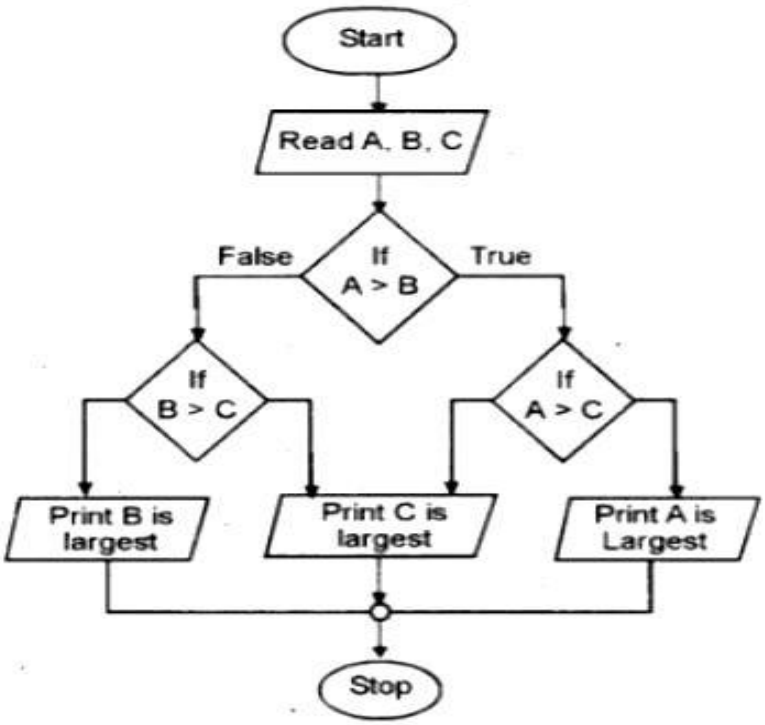
6. Draw a flowchart to find factorial of given number.(4 M)



8. Draw a flowchart to print Fibonacci series. (4 M)
Ans:

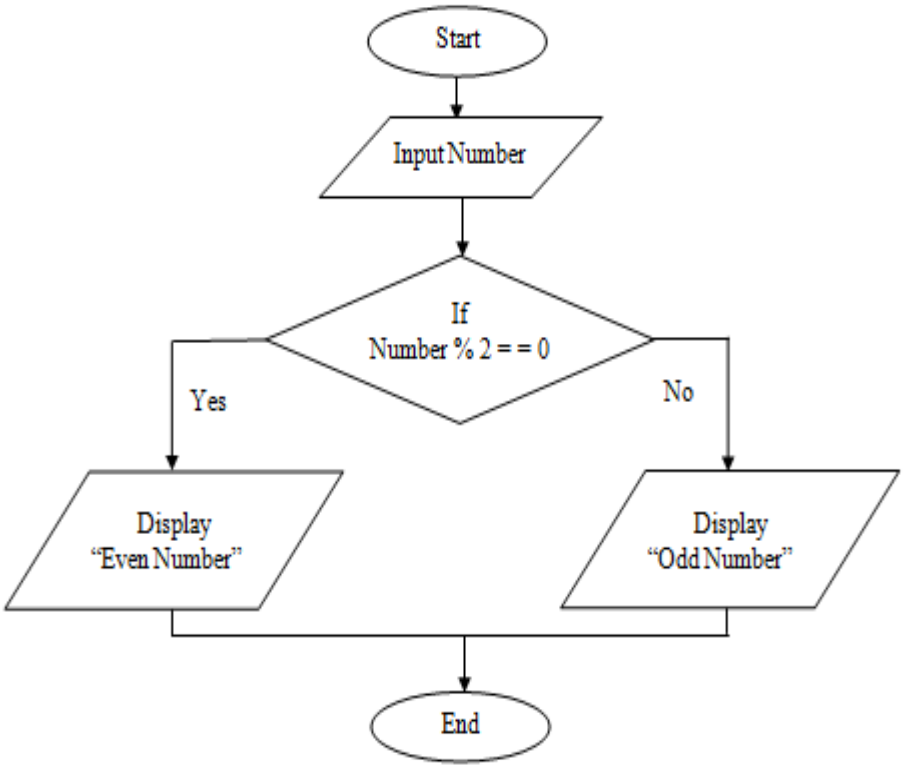


8. Draw a flowchart to find maximum out of three numbers. (4 M)
Ans:



9. Draw a flowchart to display that given number is odd or even. (4 M)

Ans:



10. Write an algorithm to find square of given number. (4 M)

Ans: The algorithm can be written as:

Step 1 – start the process

Step 2 – get the input x

Step 3 – calculate the square by multiplying the input value ie., square ← x* x Step 4 – display the result square

Step 5 – Stop

11. Write an algorithm to find area of rectangle. (4 M)

Ans: The algorithm can be written as:

Step 1: Start

Step 2: Input length and breadth

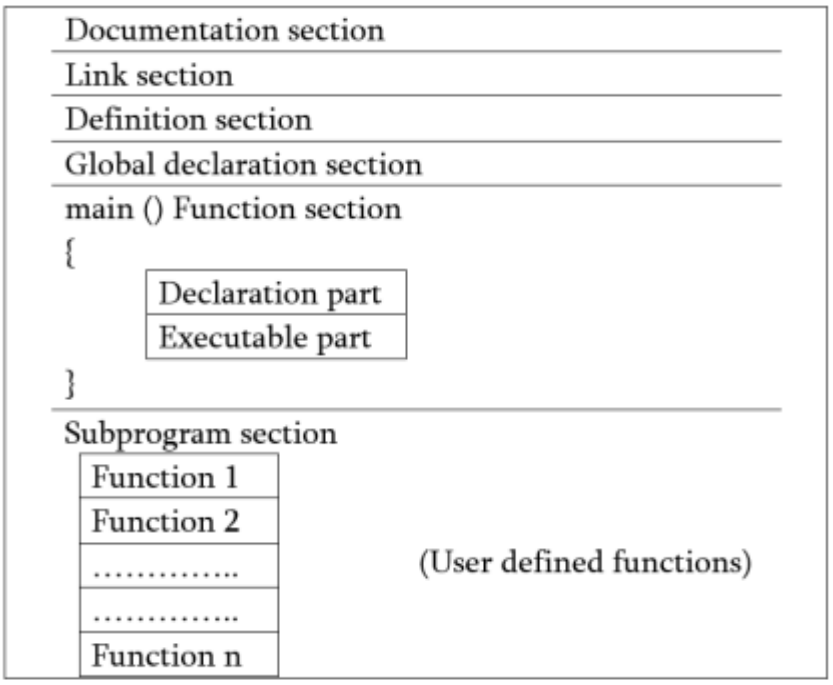
Step 3: area = length * breadth

Step 4: print area

Step 5: stop.

UNIT-2- Basics of 'C'

1.Explain basic structure of C Program. (3 M)
Ans: Basic structure of C Program:



1. Documentation section: The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.
2. Link section: The link section provides instructions to the compiler to link functions from the system library such as using the #include directive.
3. Definition section: The definition section defines all symbolic constants such using the #define directive.
4. Global declaration section: There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.
5. MAIN FUNCTION SECTION : It tells the compiler where to start the execution from
main()
{ point from execution starts }

⇒ **Main function has two sections**

1. declaration section : In this the variables and their data types are declared.
2. Executable section : This has the part of program which actually performs the task we need.

2. Write advantages of C language. (2 M)

Ans: List out advantages of C Language:

1. It is easy to understand
2. Presence of many Libraries
3. Easy to write
4. Low cost
5. Fast execution speed
6. Portable
7. Easy debugging
8. Procedure Oriented Language
9. Speed of Compilation
10. Execution of algorithms and data structures
11. Dynamic memory allocation.

3. Explain Tokens of C language. (2 M)

Ans: Tokens in C language can be divided into the following categories:

- Keywords in C
- Identifiers in C
- Strings in C
- Operators in C
- Constant in C
- Special Characters in C

Keywords in C:
Keywords in C can be defined as the pre-defined or the reserved words having its own importance, and each keyword has its own functionality.
Identifiers in C

Identifiers in C:
Identifiers in C are used for naming variables, functions, arrays, structures, etc. Identifiers in C are the user-defined words.

Strings in C:
Strings in C are always represented as an array of characters having null character '\0' at the end of the string.

Operators in C:
Operators in C is a special symbol used to perform the functions. The data items on which the operators are applied are known as operands. Operators are applied between the operands.

Unary Operator:
A unary operator is an operator applied to the single operand. For example: increment operator (++), decrement operator (--), sizeof, (type)*.

Binary Operator:

The binary operator is an operator applied between two operands.

Constants in C:
A constant is a value assigned to the variable which will remain the same throughout the program, i.e., the constant value cannot be changed.

Special characters in C:
Some special characters are used in C, and they have a special meaning which cannot be used for another purpose.

4. What is declaration & initialization of variable? Give Example. (4 M)

Ans:

- Declaration of a variable in a computer programming language is a statement used to specify the variable name and its data type. Declaration tells the compiler about the existence of an entity in the program and its location. When you declare a variable, you should also initialize it.
- Initialization is the process of assigning a value to the Variable. Every programming language has its own method of initializing the variable. If the value is not assigned to the Variable, then the process is only called a Declaration.

The basic form of declaring a variable is:

type identifier [= value] [, identifier [= value]]…];

OR

data_typevariable_name = value;

5.Write the rules for defining a variable. (2 M)
Ans: Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, goto , etc.

6.Write a C program to print "Hello World" on your screen. (3 M)

Ans:
```
#include<stdio.h>
int main()
{
printf("Hello World");
return0;
}
```
Output:
HelloWorld

7. Explain type conversion with example. (4 M)

Ans: A type cast is basically a conversion from one type to another.

There are two types of type conversion:

1. **Implicit Type Conversion**
2. **ExplicitType Conversion**

Implicit Type Conversion:

- It is also known as 'automatic type conversion'.
- Done by the compiler on its own, without any external trigger from the user.
- Generally takes place when in an expression more than one data type is present.
- In such condition type conversion (type promotion) takes place to avoid loss of data.
  bool -> char -> short int -> int ->
  unsigned int -> long -> unsigned ->
  longlong -> float -> double -> long double

It is possible for implicit conversions to lose information, signs can be lost.

Example of Type Implicit Conversion:

```
#include<stdio.h>
int main()
{
int x = 10;    // integer x
char y = 'a';  // character c

   // y implicitly converted to int. ASCII
   // value of 'a' is 97
   x = x + y;
   // x is implicitly converted to float
float z = x + 1.0;
printf("x = %d, z = %f", x, z);
return 0;
}
```

Output:

```
x = 107, z = 108.000000
```

ExplicitType Conversion:

- This process is also called type casting and it is user defined. Here the user can type cast the result to make it of a particular data type.

The syntax in C:

(type) expression

Example:

```
#include<stdio.h>
int main(){
double x = 1.2;
int sum = (int)x + 1;
printf("sum = %d", sum);
return 0;}
```

Output:
sum = 2

8.Write a C program to find area of circle using Constant. (4 M)
Ans:
```
#include<stdio.h>
int main()
{
const float pi=3.14;
int radius;
float area;
printf("Enter Radius:");
scanf("%d",&radius);
area=pi * (radius * radius);
printf("Area of Circle: %.2f",area);
return 0;
}
```

9.Explain keywords of C. (2 M)

Ans: A keyword is a reserved word. You cannot use it as a variable name, constant name, etc. There are only 32 reserved words (keywords) in the C language.

A list of 32 keywords in the c language is given below:

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

10.Write to program to check that given number is positive or negative. (4 M)

Ans:
```
#include <stdio.h>
void main()
{
intnum;
printf("Enter a number: \n");
scanf("%d", &num);
if (num> 0)
printf("%d is a positive number \n", num);
else if (num< 0)
printf("%d is a negative number \n", num);
else
printf("0 is neither positive nor negative");
}
```

Output :

```
Enter a number:
0
0 is neither positive nor negative
```

11.Write a program to find given number is odd or even (4 M)

Ans:
```
#include <stdio.h>
int main() {
intnum;
printf("Enter an integer: ");
scanf("%d", &num);
if(num % 2 == 0)
printf("%d is even.", num);
else
printf("%d is odd.", num);
return 0;
}
```

Output:

```
Enter an integer: -7
-7 is odd.
```

12.What is identifiers? (2 M)

Ans: C IDENTIFIERS Identifiers are used as the general terminology for the names of variables, functions and arrays.

There are certain rules that should be followed while naming c identifiers:

- They must begin with a letter or underscore (_).
- They must consist of only letters, digits, or underscore.

- No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only first 31 characters are significant.

13.Explain the types of constant. (2 M)

Ans: There are maintwo types of Constant:

1. NUMERIC CONSTANTS:
2. CHARACTER CONSTANTS:

- **NUMERIC CONSTANTS:**

1. integer constant is 786,-127
2. Long constant is written with a terminal 'l'or 'L',for example 1234567899L is a Long constant.
3. Unsigned constants are written with a terminal 'u' or 'U', and the suffix 'ul' and 'UL' indicates unsigned long.
4. Floating point constants contain a decimal point or an exponent or both.

- **CHARACTER CONSTANTS:**

A character constant is written as one character with in single quotes such as 'a'. The value of a character constant is the numerical value of the character in the machines character set.

The following are the some of the examples of escape sequences:

Escape sequence	**Description**
\a	Alert
\b	Backspace
\f	Form feed
\n	New Line
\r	Carriage return
\t	Horizontal Tab
\v	Vertical Tab

String constants: is a sequence of zero or more characters surrounded by a double quote.

Enumeration constant : it is a list of constant integer values.

Ex.: enum color { RED, Green, BLUE } The first name in the enum has the value 0 and the next 1 and so on unless explicit values are specified. If not all values specified , unspecified values continue the progression from the last specified value.

14. Explain with an example of dynamic initialization of variables. (4 M)

Ans: Dynamic initialization of object refers to initializing the objects at run time i.e. the initial value of an object is to be provided during run time. Dynamic initialization can be achieved using constructors and passing parameters values to the constructors. This type of initialization is required to initialize the class variables during run time.

Dynamic initialization of objects is needed as:

1. It utilizes memory efficiently.
2. Various initialization formats can be provided using overloaded constructors.
3. It has the flexibility of using different formats of data at run time considering the situation.

15. Explain modifiers in C. (2 M)

Ans: It specifies the amount of memory space to be allocated for a variable. Modifiers are prefixed with basic data types to modify the memory allocated for a variable.

There are five data type modifiers in C Programming Language:

- long
- short
- signed
- unsigned
- long long

Character : **Character data type** is used to store a character. A variable of character data type allocated only one byte of memory and can store only one character. Keyword char is used to declare variables of type character. For Example: char ch = 'A';

Integer : **Integer data type** is used to store a value of numeric type. Keyword int is used to declare variables of integer type. Memory size of a variable of integer data type is dependent on Operating System. For Example: int count = 10;

Float : **Floating point data type** is used to store a value of decimal values. Keyword float is used to declare variables of floating data type. For Example: float rate = 5.6;

Double : **Double data type** is similar to floating data type except it provides up-to ten digit of precision and occupies eight bytes of memory. For Example: double d = 11676.2435676542;

UNIT-3 OPERATOR AND EXPRESSIONS

1. What is operator? List various operators available in c. (2 M)

Ans: An operator is a symbol that tells the compiler to perform specific mathematical or logical functions.

Types of operators −

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

2. Explain Arithmetic operators with examples. (4 M)

Ans: Arithmetic operator Performs Mathematical operation such as addition, subtraction, multiplication, division, increment operator, decrement operator.

For Example:

```
#include <stdio.h>
main() {
int a = 21;
int b = 10;
int c ;
   c = a + b;
printf("Line 1 - Value of c is %d\n", c );
   c = a - b;
printf("Line 2 - Value of c is %d\n", c );
   c = a * b;
printf("Line 3 - Value of c is %d\n", c );
   c = a / b;
printf("Line 4 - Value of c is %d\n", c );
   c = a % b;
printf("Line 5 - Value of c is %d\n", c );
   c = a++;
printf("Line 6 - Value of c is %d\n", c );
   c = a--;
printf("Line 7 - Value of c is %d\n", c );
}
```

Output:

```
Line 1 - Value of c is 31
Line 2 - Value of c is 11
Line 3 - Value of c is 210
Line 4 - Value of c is 2
Line 5 - Value of c is 1
Line 6 - Value of c is 21
Line 7 - Value of c is 22
```

3. Explain Relational operators with examples. (4 M)

Ans: The relational operators supported by C language. Assume variable A holds 10 and variable B holds 20 then −

- **==** **Checks if the values of two operands are equal or not.** If yes, then the condition becomes true. (A == B) is not true.
- **!=** **Checks if the values of two operands are equal or not.** If the values are not equal, then the condition becomes true. (A != B) is true.
- **>** **Checks if the value of left operand is greater than the value of right operand**. If yes, then the condition becomes true. (A > B) is not true.
- **<** **Checks if the value of left operand is less than the value of right operand.** If yes, then the condition becomes true. (A < B) is true.
- **>=** **Checks if the value of left operand is greater than or equal to the value of right operand**. If yes, then the condition becomes true. (A >= B) is not true.
- **<=** **Checks if the value of left operand is less than or equal to the value of right operand.** If yes, then the condition becomes true. (A <= B) is true.

For Example:

```
#include <stdio.h>
Main() {
Int a = 21;
Int b = 10;
Int c ;
If( a == b ) {
Printf("Line 1 - a is equal to b\n" );
  } else {
Printf("Line 1 - a is not equal to b\n" );  }
If ( a< b ) {
Printf("Line 2 - a is less than b\n" );
  } else {
Printf("Line 2 - a is not less than b\n" );
  }=
If ( a> b ) {
Printf("Line 3 - a is greater than b\n" );
  } else {
Printf("Line 3 - a is not greater than b\n" );}
  /* Lets change value of a and b */
A = 5;
B = 20;
If ( a<= b ) {
Printf("Line 4 - a is either less than or equal to  b\n" );
  }
If ( b>= a ) {
Printf("Line 5 - b is either greater than  or equal to b\n" );
```

```
  }
}
```

Output:

```
Line 1 - a is not equal to b
Line 2 - a is not less than b
Line 3 - a is greater than b
Line 4 - a is either less than or equal to  b
Line 5 - b is either greater than  or equal to b
```

4. Explain Logical operators with example. (4 M)

Ans: The logical operators supported by C language. Assume variable A holds 1 and variable B holds 0.

1. **&&Called Logical AND operator.** If both the operands are non-zero, then the condition becomes true. (A && B) is false.
2. **||** **Called Logical OR Operator**. If any of the two operands is non-zero, then the condition becomes true. (A || B) is true.
3. **!** **Called Logical NOT Operator.** It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false. !(A && B) is true.

For Example:

```
#include <stdio.h>
Main() {
Int a = 5;
Int b = 20;
Int c ;
If ( a&& b ) {
Printf("Line 1 - Condition is true\n" );}
If ( a || b ) {
Printf("Line 2 - Condition is true\n" );   }
A = 0;
B = 10;
If ( a&& b ) {
Printf("Line 3 - Condition is true\n" );
  } else {
Printf("Line 3 - Condition is not true\n" );
  }
If ( !(a && b) ) {
Printf("Line 4 - Condition is true\n" );
  }
}
```

Output:

```
Line 1 - Condition is true
Line 2 - Condition is true
Line 3 - Condition is not true
```

Line 4 - Condition is true

5. Explain Assignment operators with example. (4 M)

Ans: The following table lists the assignment operators supported by the C language −

1. **=** **Simple assignment operator**. Assigns values from right side operands to left side operand, For Example:C = A + B will assign the value of A + B to C
2. **+=** **Add AND assignment operator.** It adds the right operand to the left operand and assign the result to the left operand. For Example:C += A is equivalent to C = C + A
3. **-=** **Subtract AND assignment operator.** It subtracts the right operand from the left operand and assigns the result to the left operand. For Example:C -= A is equivalent to C = C – A
4. ***=** **Multiply AND assignment operator.** It multiplies the right operand with the left operand and assigns the result to the left operand. For Example:C *= A is equivalent to C = C * A
5. **/=** **Divide AND assignment operator.** It divides the left operand with the right operand and assigns the result to the left operand. For Example /= A is equivalent to C = C / A

%= **Modulus AND assignment operator.** It takes modulus using two operands and assigns the result to the left operand. For Example:C %= A is equivalent to C = C % A

<<=Left shift AND assignment operator. For Example:C <<= 2 is same as C = C << 2

>>=Right shift AND assignment operator. For Example:C >>= 2 is same as C = C >> 2

&= **bitwise AND assignment operator.**For Example:C &= 2 is same as C = C & 2

^= **Bitwise exclusive OR and assignment operator**. For Example:C ^= 2 is same as C = C ^ 2

|= **Bitwise inclusive OR and assignment operator.** For Example:C |= 2 is same as C = C | 2

Example:

```
#include <stdio.h>
main() {
int a = 21;
int c ;
  c =  a;
printf("Line 1 - =  Operator Example, Value of c = %d\n", c );
  c +=  a;
printf("Line 2 - += Operator Example, Value of c = %d\n", c );
  c -=  a;
printf("Line 3 - -= Operator Example, Value of c = %d\n", c );
```

```
c *=  a;
printf("Line 4 - *= Operator Example, Value of c = %d\n", c );
c /=  a;
printf("Line 5 - /= Operator Example, Value of c = %d\n", c );
c  = 200;
c %=  a;
printf("Line 6 - %= Operator Example, Value of c = %d\n", c );
  c <<=  2;
printf("Line 7 - <<= Operator Example, Value of c = %d\n", c );
  c >>=  2;
printf("Line 8 - >>= Operator Example, Value of c = %d\n", c );
c&=  2;
printf("Line 9 - &= Operator Example, Value of c = %d\n", c );
c ^=  2;
printf("Line 10 - ^= Operator Example, Value of c = %d\n", c );
c |=  2;
printf("Line 11 - |= Operator Example, Value of c = %d\n", c );
}
```

Output:

```
Line 1 - =  Operator Example, Value of c = 21
Line 2 - += Operator Example, Value of c = 42
Line 3 - -= Operator Example, Value of c = 21
Line 4 - *= Operator Example, Value of c = 441
Line 5 - /= Operator Example, Value of c = 21
Line 6 - %= Operator Example, Value of c = 11
Line 7 - <<= Operator Example, Value of c = 44
Line 8 - >>= Operator Example, Value of c = 11
Line 9 - &= Operator Example, Value of c = 2
Line 10 - ^= Operator Example, Value of c = 0
Line 11 - |= Operator Example, Value of c = 2
```

6. Explain conditional operators with example. (4 M)

Ans: The conditional operator is also known as a ternary operator. The conditional statements are the decision-making statements which depends upon the output of the expression. It is represented by two symbols, i.e., '?'

Syntax :

(condition)?expression 1:expression2;

For Example:

```
#include<stdio.h>
int main()
{
int m = 5, n = 4;
   (m > n) ?
printf("m is greater than n that is %d > %d",m, n):
printf("n is greater than m that is %d > %d",n, m);
return 0;
}
```

Output:

m is greater than n that is 5

7. Explain Bitwise operators with example. (4 M)

Ans: In C, 6 operators are bitwise operators (work at bit-level)

The & (bitwise AND) in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

The | (bitwise OR) in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.

The ^ (bitwise XOR) in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

The << (left shift) in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.

The >> (right shift) in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.

The ~ (bitwise NOT) in C or C++ takes one number and inverts all bits of it

For Example:

```
#include <stdio.h>
int main()
{
   // a = 5(00000101), b = 9(00001001)
unsigned char a = 5, b = 9;
   // The result is 00000001
printf("a = %d, b = %d\n", a, b);
printf("a&b = %d\n", a & b);
   // The result is 00001101
printf("a|b = %d\n", a | b);
   // The result is 00001100
```

```
printf("a^b = %d\n", a ^ b);
     // The result is 11111010
printf("~a = %d\n", a = ~a);
     // The result is 00010010
printf("b<<1 = %d\n", b << 1);
     // The result is 00000100
printf("b>>1 = %d\n", b >> 1);
return 0;
   }
```

Output:

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>>1 = 4
```

8. Write a program to design a calculator using arithmetic operator. (4 M)

Ans:

```
#include <stdio.h>
int main()
{
   // declare local variables
char opt;
int n1, n2;
float res;
printf (" Select an operator (+, -, *, /) to perform an operation in C calculator \n ");
scanf ("%c", &opt); // take an operator
printf (" Enter the first number: ");
scanf(" %d", &n1); // take fist number
printf (" Enter the second number: ");
scanf (" %d", &n2); // take second number
if (opt == '+')
   {
res = n1 + n2; // add two numbers
printf (" Addition of %d and %d is: %f", n1, n2, res);
   }
else if (opt == '-')
   {
res = n1 - n2; // subtract two numbers
printf (" Subtraction of %d and %d is: %f", n1, n2, res);
   }
else if (opt == '*')
   {
res = n1 * n2; // multiply two numbers
```

```
printf (" Multiplication of %d and %d is: %f", n1, n2, res);
   }
else if (opt == '/')
   {
if (n2 == 0)   // if n2 == 0, take another number
      {
printf (" \n Divisor cannot be zero. Please enter another value ");
scanf ("%d", &n2);
      }
res = n1 / n2; // divide two numbers
printf (" Division of %d and %d is: %.2f", n1, n2, res);
   }
else
   {
printf(" \n You have entered wrong inputs ");
   }
return 0;
}
```

Output:

```
Select an operator (+, -, *, /) to perform an operation in C calculator
/
Enter the first number: 20
Enter the second number: 0

Divisor cannot be zero. Please enter another value 5
Division of 20 and 5 is: 4.00
```

9. Write a program to swapping two numbers using Bitwise Operator. (4 M)

Ans:

```
#include<stdio.h>
int main(){
inta,b;
printf("enter the values for a and b:");
scanf("%d%d",&a,&b);
printf("value of a=%d and b=%d before swap\n",a,b);
  a= a^b;
  b= a^b;
  a= a^b;
printf("value of a=%d and b=%d after swap",a,b);
return 0;
}
```

Output :

```
enter the values for a and b:24 56
value of a=24 and b=56 before swap
value of a=56 and b=24 after swap
```

UNIT-4- DECISION STATEMENT

1. Explain simple If statements. (2 M)

Ans: General Syntax of simple if statement is as follow:

- **Syntax:**

If (expression)

{

Statement :

Statement 2;

}

Statement - X;

- In the above syntax the statement block may be a single statement or a group of statements.
- The simple if statement is executed in the following order.

1. first the condition is checked.
2. if the condition is true then the statement block is executed and then statement-x is executed.

- if the condition is false then only statement –x is executed.

2. Explain If….Else statements. (3 M)

Ans:

- The general syntax of if..else statement is as follow:
- Syntax:

if (test expression)

{

True-block statements;

}

else

{

False-block statements;

}

- The if..else statement is executed in the following order:

1. first the condition is checked.
2. if condition is true the true statement is executes.
3. if condition is false the false statement is executed.

3. Explain Nested if-else statement with example. (4 M)

Ans:

- When the more then one condition is to be checked then we can use nesting of if..else first the condition is checked.
- The syntax of nesting if..else statement is as follow:

Syntax:

```
if (test condition 1)
{
if (test condition 2)
{
true-block2 statements;
}
else
{
false-block2 statements;
}
else
{
false-block1 statements;
}
```

- The nesting if-else statement is executed in the following manner.

1. First the condition 1 is checked.
2. if the condition 1 is true then condition 2 is checked. If condition 2 is true then statement-1 is executed.

3. but if the condition 2 is false the statement-2 is executed.
4. if condition 1 is false the statement-3 is executed.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num1, num2, num3;
    printf("Enter three numbers:\n");
    scanf("%d%d%d",&num1, &num2, &num3);
    if(num1>num2)
    {
     /* This is nested if-else */
     if(num1>num3)
     {
            printf("Largest = %d", num1);
     }
     else
     {
            printf("Largest = %d", num3);
     }
    }
    else
    {
     /* This is nested if-else */
     if(num2>num3)
     {
            printf("Largest = %d", num2);
     }
     else
     {
```

```
            printf("Largest = %d", num3);
        }
       }
       return(0);
}
```

4. Explain If else-if ladder with example. (4 M)

Ans:

- The if..else ladder statement provide two-way decision where we select one of the alternative.
- It is used for multiple choice.
- The two way decision is done by nested if..else is not sufficient.
- Following is the syntax of the if..else ladder.

Syntax:

```
if (condition 1)
statement 1;
else if (condition 2)
statement 2;
else if (condition 3)
statement 3;
else if (condition n)
statement n;
else
default statement;
}
```

- if-else-if ladder is executed in the following order:

1. first condition-1 is executed , if the condition-1 is true then the statement-1 is executed.
2. if the condition-1 is false then condion-2 is checked. If condition-2 is true then statement-2 is executed.

3. This procedure repeated until all the condition is checked. if all the condition became false then the default statement is executed.

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a,b,c;
    printf("Enter three numbers: \n");
    scanf("%d%d%d", &a, &b, &c);
    if(a>b && a>c)
    {
        printf("Largest = %d", a);
    }
    else if(b>a && b>c)
    {
        printf("Largest = %d", b);
    }
    else
    {
        printf("Largest = %d", c);
    }
    return(0);
}
```

5. Explain Switch …case statement with example. (4 M)

Ans:

- The switch statement is also known as multi-choice or multi decision statement.
- Writing the code using the multiple if..else becomes lengthy and also difficult to manage. Using switch statement it is done by easy.
- It provide the choice for each value of variable or expression.
- The switch statement test the value of a given variable against a list of case values and when the match is found, a statement associated with the case is executed.

- Syntax:

```
Switch(variable name or expression)
switch (year) {
case const-expr 1: statement 1;
break;
case const-expr 2: statement 2;
break;
case const-expr3: statement 3;
break;
default:
Statements;
}
```

- The expression or variable name is an integer or characters.
- Value-1,value-2 is constant or known as case labels ,each case lable value must be unique with a switch statement. each case lable must end with (;).
- The switch..case statement is executed in the following order:

1. the value of the expression is compared against the value of case lable.
2. if the case is found whose value match with value of the expression ,then the statement associated with the case is executed. there is a single statement or multiple statement. There is break which send the control to the next statement.
3. if the value of the expression does not match with any case value then the

statement associated with the default case is executed.

Example:

```
#include <stdio.h>
#include<conio.h>
int main()
{
    int num=2;
    switch(num+2)
    {
       case 1:
         printf("Case1: Value is: %d", num);
       case 2:
         printf("Case1: Value is: %d", num);
       case 3:
         printf("Case1: Value is: %d", num);
       default:
         printf("Default: Value is: %d", num);
   }
   return 0;
}
```

6. Explain Break statement with example. (4 M)

Ans:

- We have use the break statement with the switch statement.
- The function of of break statement is exit form the switch body.
- If it is not written after each case statement, then control pass to the next statement ,so remaining statement of the next case statement will also execute even if the case value do not match and the program will not function properly.

Example:

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

```
{
 int i;
 for(i=10;i>0; i--)
 {
  if(i==6)
   {
     printf("\n Coming out from for loop Where i = %d\n", i);
     break;
   }
  printf(" %d  ",i);
 }}
```

7. What is the difference Between If…Else and Switch case. (4 M)

Ans:

IF-ELSE	SWITCH
If statement is used to select among two alternatives	The switch statement is used to select among multiple alternatives.
If can have values based on constraints.	Switch can have values based on user choice.
If implements Linear search.	Switch implements Binary search.
Float, double, char, int and other data types can be used in if condition.	Only int and char data types can be used in switch block.
It is difficult to edit the if-else statement, if the nested if-else statement is used.	It is easy to edit switch cases as, they are recognized easily.

8. Give the difference between Break & Continue. (4 M)

Ans:

Break Statement	Continue Statement
The Break statement is used to exit from the loop constructs.	The continue statement is not used to exit from the loop constructs.
The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
When a break statement is encountered then the control is exited from the loop construct immediately.	When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.
Syntax: break;	**Syntax:** continue;

9. Write a program to find given no is odd or even. (4 M)

Ans:

```
#include <stdio.h>
#include<conio.h>
void main()
 {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   if(num % 2 == 0)
      printf("%d is even.", num);
```

```
    else
        printf("%d is odd.", num);
        return 0;
}
```

10. Write a program to display days using switch case. (4 M)

Ans:

```
#include <stdio.h>
 #include<conio.h>
void main() {
    int day;
    printf("Enter Day Number (1 = Monday ..... 7 = Sunday)\n");
    scanf("%d", &day);
     switch(day){
        case 1 : printf("Monday\n");
            break;
        case 2 : printf("Tuesday\n");
            break;
        case 3 : printf("Wednesday\n");
            break;
        case 4 : printf("Thursday\n");
            break;
        case 5 : printf("Friday\n");
            break;
        case 6 : printf("Saturday\n");
            break;
        case 7 : printf("Sunday\n");
            break;
        default: printf("Invalid Input !!!!\n");
    }
     return 0;
}
```

11. Write a program to find if a given number is positive or negative. (4M)

Ans:

```
#include <stdio.h>
#include<conio.h>
void main() {
   double num;
   printf("Enter a number: ");
   scanf("%lf", &num);
   if (num <= 0) {
      if (num == 0)
         printf("You entered 0.");
      else
         printf("You entered a negative number.");
   }
   else
      printf("You entered a positive number.");
   return 0;
}
```

UNIT- 5 LOOP CONTROL STATEMENTS

1. Define loop control statements and list its types. (3 M)

Ans:

Loops are to repeat execution of block of code.

During looping a set of statements are executed until some condition for termination is uncounted.

General loop process would include the following four steps:

I. Initialization of counter
II. Test for a termination condition
III. Loop body statements
IV. Increment the counter

- Depending on where the condition is checked, we can have two types of loop structure:

1. Entry control loop.
2. exit control loop.

- Entry control loop: the condition is written first and then the body of statement. If the condition is tested before the body of loop is called Entry control loop.if condition is true then the body of loop is executed otherwise the loop is not executed.
- Exit control loop:the body of statement is written first then the condition is written. If The condition condition is tested after the body of the loop then it is known as exit control loop. So first body of the loop is executed and then the condition is checked.

2. Write a c program to print sum of first 10 nos. Using for loop. (4 M)

Ans:

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

   printf("Enter a positive integer: ");
   scanf("%d", &n);

   for (i = 1; i <= n; ++i) {
      sum += i;
   }

   printf("Sum = %d", sum);
   return 0;
}
```

3. State advantages of using nested for loop. (3 M)
Ans:

- A nested for loop is a loop nut it has a loop inside another loop. simply a loop inside other loop is called as nested loop.
- It is useful when we are dealing with more number of iterations.
- Reduces that number of lines of program
- reusing of code is possible
- memory size usage will be reduced.

4. Differentiate while and do while loop. (3 M)
Ans:

While Loop	**Do… While Loop**

while (condition) {
statements; //body of loop
}
	do{
	.
	statements; // body of loop.
	.
	} while(Condition);

In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
The iterations do not occur if, the condition at the first iteration, appears false.	The iteration occurs at least once even if the condition is false at the first iteration.
Entry-controlled loop	Exit-controlled loop

5. Explain WHILE LOOP with Example. (4 M)
Ans:

• While loop is the entry control loop. Because in while loop first the condition is checked.
• Following is the syntax of the while loop.
• Syntax:

```
While(condition)
{
Body of the loop
}
```

• {} is known as body of the loop. If body contain one statement then it is not necessary to enclose body of the loop in the pair of brace {}.
• The while loop is executed in the following format.
• Here the condition is evaluated first and if it is true then the statement in

the body of the loop is executed.

- After executing body , the condition is evaluated again and if it is true body is executed again. This process is repeated as long as the condition is true. The control move out once the condition is false.

Example:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int count, limit;
printf ("Enter your age: " ) ;
scanf ("%d", &limit ) ;
while (count<=limit)
{
printf (" %d", count);
count=count+1;
}
getch();
}
```

6. Explain DO…WHILE LOOP with example. (4 M)

Ans:

- Do..while loop is a exit control loop.
- Following is the syntax of the exit control loop. Because after the executing the body of the loop the condition is checked.
- Syntax:

```
Do
{
Body of the loop
}while(condition);
```

- The do while loop is executed in the following format.
- In do..while loop the body is executed and then the condition is checked.
- If the condition is true the body is executed again and again, as long as the condition is true.
- The control moves out of loop once, the condition become false.
- The do..while loop is exit control loop so first body is executed the the condition is checked. this ensure that the body of the loop is executed at lease once even if the condition is false first time.

Example:

```
#include <stdio.h>
int main()
{
double number, sum = 0;
do
{
```

```
printf("Enter a number: ");
scanf("%lf", &number); sum += number;

}
while(number != 0.0);
printf("Sum = %.2lf",sum);
return 0;
}
```

7. Write a c program using nested for loops to find prime numbers between 2 to 100. (4 M)
Ans:

```
#include<stdio.h>
#include<math.h>
 void main()
{
   int start, end, num, count, prime, temp, inum;
     printf("Enter start and end value\n");
   scanf("%d%d", &start, &end);
     if(start > end)
   {
      temp = start;
      start= end;
      end  = temp;
   }
     printf("Prime Numbers between %d and %d are\n", start, end);
     for(num = start; num <= end; num++)
   {
      prime = 1;
      inum  = sqrt(num);
      for(count = 2; count <= inum; count++)
      {
         if(num % count == 0)
         {
            prime = 0;
            break;
         }
      }
        if(prime) printf("%d,\t", num);
   }
     return 0;
}
```