Questions:

PPS (UNIT-1)

Short Answer Questions:

1. Define an Operating System.
2. Write short notes on programming languages.
3. Differentiate between low-level and high level languages.
4. Differentiate between an assembler and translator.
5. Differentiate between Interpreter and compiler.
6. List various data types used in C language.
7. Explain C-Strnt and C-expression in C language.

Long Answer Questions:

1. Block Diagram of a Computer.
2. Steps in writing and executing C Program.
3. Algorithm and flow chart
   i) Max & Min of 2 No.s ii)
   Max & Min of 3 No.s iii)
   To check whether a no. is
   prime or not.
   iv) Factorial of a number.
4. Operators used in C language.

(UNIT-2)

Short Answer Questions:

1 Differentiate between while and do-while loop.

2 Conditional / terinary Operators.

3 Bitwise Operators.

4 Differentiate between break, continue, goto and exit strnts.

Long Answer Questions:

1 Switch Strnt with example.

2 Programs:

i) To check whether a given no is palindrome or
not.

ii) Sin (x) and Cos(x) using series expansion. iii) Sum of the series upto a given limit. iv) Armstrong no. v) Strong no.

3 Program to find roots of quadratic equation.

4 Reading display elements of a ID array.

5 Searching Techniques - * linear
  * Binary

6 Sorting Techniques- * Selection Sort
  * Bubble Sort

7 To print array elements in reverse order.

8 Matrix addition & multiplication

9 Identity Matrix

10 Symmetric Matrix

11 Muging 2 arrays into a 3rd array.

(UNIT-3)

Short Answer Questions:

1 Read and display elements of a multi dimensional array.

2 Function prototype.

3 Difference between function declaration & function definition.

4 Syntax of defining a function.

5 Parametic pauing Tecgniques.

Long Answer Questions:

1 String handling functions-{ strlen(), strcat(), strcmp(), strcat(), strrev()-etc} with and without using library function.

2 To check whether a given string is palindrome or not.

3 To arrange a given string in alphabetical order.

4 Intec function communication.

(UNIT-4)

Short Answer Questions:

1 Define recursion with an example.

2 Difference between recursion and non-recursion functions.

3 Define a structure with an example.

4 Operations performed on structures.

Long Answer Questions:

1 write a recursive function to
   i) Find factorial
   ii) GCD
   iii) Primary search using recursion iv) Fabonacci series
   v) X^n

2 3 Types of Structures - Structures with in structures
   - array of structures
   - arrays with in structures.

3 Explain how the members of the structure are accused 4 Program to Print

student details using pointer to structure.

5 Program to illustrate self-referential structure.

6 Differences between Arrays and Structures.

(UNIT-5)

Short Answer Questions:

1 Define a Pointer variable.
2 Define a file in C language.
3 Syntax of fopen()
4 Error handling functions in files.
5 Types of files in C language.
6 What are input & output Streams.
7 Various arithmetics performed on pointer variables.
8 Modes of a file.

Long Answer Questions:

1. Programs:
i) To access the elements of ID array
using pointer variable.
ii) Bubble Sort using pointers. iii)
Selection Sort using pointer. iv)
Addition of 2 matrices using

pointers. v) Matrix Multiplication
using pointers.

2 Call by reference – swapping example.
3 Dynamic memory allocation
4 File handling functions
5 To count no. of characters, words and files in a file.
6 Copy contents of one file into another file.
7 Compare the contents of two files.
8 Merging of 2 files into another file.

Answers:

PPS (UNIT-1)

Short Answer Questions:

1. Define an Operating System.

Ans: ***OPERATING SYSTEM:***

It is the interface between the user and the computer hardware. These programs are in-built into the computer and are used to govern the control of the computer hardware components, such as processors, memory devices and input/output devices.

Examples: MS-DOS, Win XP,Win 2007, Solaris ,Unix , Linux, Android, IOS.

It is system software which controls the hardware and software resources of the computer and offers common services for computer programs.

2. Write short notes on programming languages.

Ans: ***PROGRAMMING LANGUAGES:***

Programming languages are broadly classified into the following three forms.

1. MACHINE LANGUAGE:

Low level language\Machine language is the first generation language which uses 0’s and 1’s.

*Advantages:

Computer can understand directly.

*Disadvantages:

1. Difficult to modify.

2. Cannot debug.

3. It is machine dependent.

4. Suitable for simple applications.

2. ASSEMBLY LANGUAGE

An assembly language, often abbreviated asm, it is a low level language in which there is very strong correspondence between the program. It is 2nd generation language. Assembly language has several advantages over machine language. These programs are easy to write and modify than machine language programs. However, assembly language is again a machine oriented language and the program has to be different for different machines.

3. HIGH LEVEL LANGUAGES:

It is a 3rd generation language. It is based on English grammatical notations and mathematical formulas.

*Advantages:

1. Easy to follow.

2. Easy to understand.

3. Easy to modify and debug.

4. Suitable for complex applications.

*Disadvantages:

1. It requires the translator programme called compiler or interpreter.

2. Execution time is more.

3. Differentiate between low-level and high level languages.

Ans:

DIFFERENCE BETWEEN THE HIGH AND THE LOW LEVEL LANGUAGE:

1. Low level language is machine readable form of program. Whereas the high level language will be in readable form.
2. Low level language are difficult to write and compile but high level languages are easy to write as well as compile.
3. Low level language are compact and require less memory space. High level language uses compilers and interpreters which requires large memory space.
4. In high level language debugging .I.e. Finding and correcting errors are easier whereas debugging in the low level language is quite difficult.
5. Low level language coding and compiling is time consuming process whereas high level language coding and compiling is much easy and takes less time to compile.

4. Differentiate between an assembler and translator.

Ans:

*ASSEMBLER:

An assembly language can be translated into a machine language by replacing the mnemonic and symbolic information by its numeric equivalent. A program which converts an assembly language program into a machine language program is called an assembler.

TRANSLATOR:

Translators are computer codes that will convert programs written in one language into a different language. Compilers, interpreters, assemblers etc. are examples of translator.

5. Differentiate between Interpreter and compiler.

Ans:

COMPILER:

A high level language can also be converted into a machine language by a program called compiler.

A compiler checks the entire user-written program (known as the source program) and, if error free, produces a complete program in machine language (known as object program). The source program is retained for possible modifications and corrections and the object program is loaded into the computer’s memory for execution.

INTERPRETER:

An interpreter does a similar job but in a different style. The interpreter (as the name implies) translates one statement at a time and, if error-free, executes the instruction. This continues till the last statement in the program has been translated and executed. Thus, the interpreter translates and executes the first instruction before it goes to the second, while a compiler translates the whole program before execution can begin.

The major differences between them are:

a) Error Correction is much simpler in the case of an interpreter.
b) Interpreter take more time for the execution of a program compared to compilers because a statement has to be translated every time the program is executed.

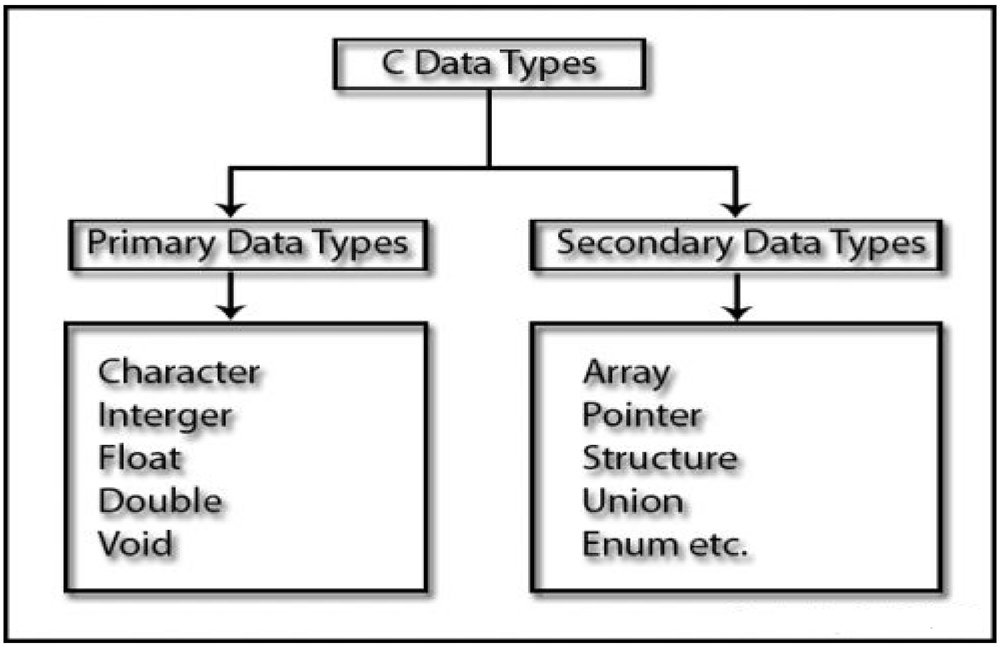
6. List various data types used in C language.

Ans:

DATA TYPES:

C language is rich in its data types. The variety of data types available allows the programmer to select the type appropriate to the needs to the application. C has different data types for different types of data and can be broadly classified as:

1. Primary data types
2. Secondary data types

3. 

7. Explain C-Statement and C-expression in C language.

Ans:

*C-Statement:

A **statement** is a command given to the computer that instructs the computer to take a specific action, such as display to the screen, or collect input. A computer program is made up of a series of statements.

Note: Every C statement must end with semicolon; this; acts as a statement terminator.

*C-expression:

An **expression** in a programming language is a combination of one or more constants, variables, operators, and functions that the programming language interprets and computes to produce another value.

These expressions are evaluated left to right and the value of right most expression is the value of the combined expression.

Ex: v=(x=10, y=5, x+y);

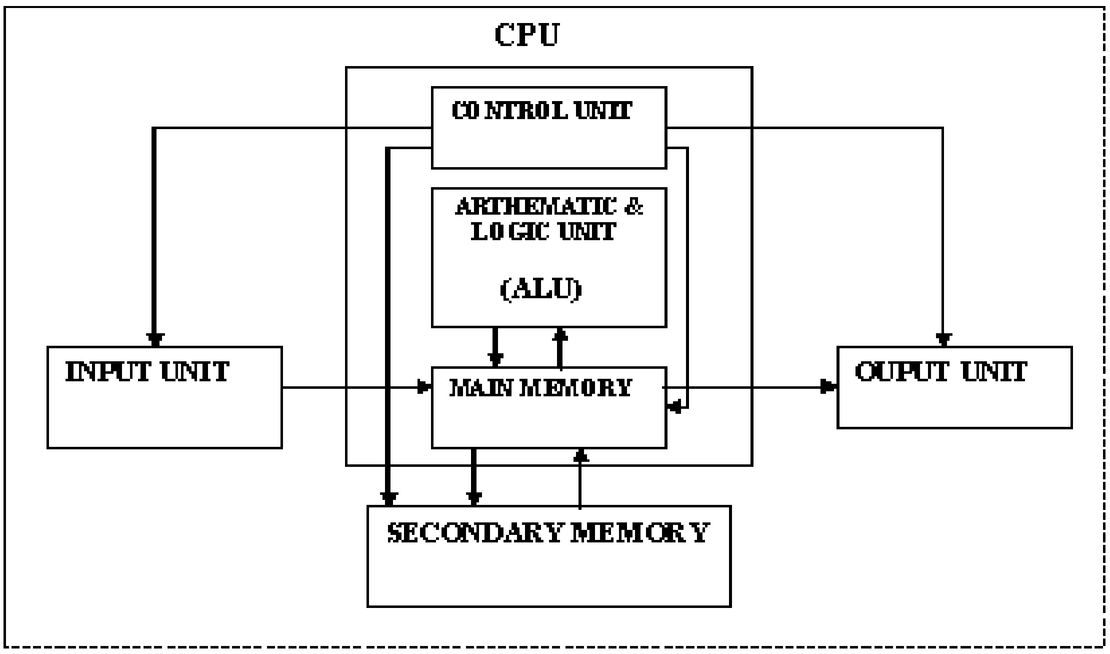
Here the value of the right most expression is stored in variable v i.e, 15.

Long Answer Questions:

1. Block Diagram of a Computer.

Ans:

BLOCK DIAGRAM OF A COMPUTER:



1. INPUT UNIT:

It obtains information (data and programs) from various input devices and place this information at the disposal of the other units so that information may be processed. Most information is entered into computer today through keyboard and mouse devices. Information can also enter by speaking and by scanning images. Some of other input devices are touch screen (used in ATM's), joystick, electronic pen etc

2. CENTRAL PROCESSING UNIT:

It is the mind and heart combined with the central nervous system of a computer. So the CPU performs functions similar to the mind, heart of a human body. It consists of three components

1) Arithmetic & Logic Unit (ALU)
2) Storage devices
   a. Main memory
   b. Secondary memory
3) Control Unit

The functions of CPU are to:-

(i) Stores data as well as instructions.
(ii) Controls the sequence of operations as per the stored instructions.
(iii) Issue commands to all parts of the computer system.
(iv) Carry out data processing and send results to output

2.1 Arithmetic and Logic Unit (A.L.U):

It operates on the data available in the main memory and sends them back after processing, once again to the main memory. A.L.U performs two functions

1) It carries out arithmetical operations like addition, subtraction, multiplication and division.
2) It performs certain logical actions based on AND and OR functions.

2.2 Control Unit:

The control unit directs all operations inside the computer. It is known as nerve centre of the computer because it controls and coordinates all hardware operations i.e. those of the CPU and input-output devices. Its actions are

1. It gives command to transfer data from the input device to the memory to ALU.
2. It also transfers the results from ALU to the memory and onto the output device for printing.
3. It stores the program in the memory, takes instructions one by one, understands them and issues appropriate commands to the other units.
4. It fetches the required instructions from the main storage.

2.3.1 Primary Memory:

It is also called as main memory because, like the human memory it is able to store information, which can be recalled or accessed when required. The program of instructions has to be stored in the main memory in order to make it work automatically.

Any item of data or any instruction stored in this memory can be retrieved by the computer at high speed. The modern computer does this in Nano seconds. Main high speed memory limited in size and very costly to buy.

Examples: RAM (Random access memory)

2.3.2 Secondary Memory:

These are also known as extended or auxiliary memory. These are the devices that hold the mass of information, which may be transferred during processing. These devices are used for permanent storage of data. As compared to the primary memory, it has a much larger capacity, but is not as fast. The computer thus takes slightly more time to retrieve from secondary storage. Secondary storage is much cheaper than main memory.

Example: Hard disk, Compact disks (CDs)

3. OUTPUT UNIT:

The result of any computer processing has to be communicated to the user. Output devices translate the computers output into a form understandable to human beings. Some of the output devices are display screen, printer etc.

HARDWARE:

It is a general term used to represent the physical and tangible components of the computer itself i.e. those components which can be touched and seen. It includes

(1) Input devices
(2) Output devices
(3) Central processing Unit
(4) Storage devices

SOFTWARE:

It is a general term to describe all the forms of programs associated with a computer. Without software, a computer is like a human body without a soul. There are

Four major categories of software

(1) Operating systems

(2) Utility programs
(3) Language processors
(4) Application programs

2. Steps in writing and executing C Program.

Ans:

STRUCTURE OF A C PROGRAM

```
Include header file section
Global declaration section
        /"*comments */
main()
{
//comments
Declaration part
Executable part
}
User defined functions
{
}
```

Include Header File Section

C program depends upon some header files for function definition that are used in program. Each header file by default is extended with .h. The header file should be included using # include directive as given here.

Global Declaration

This section declares some variables that are used in more than one function. These variables are known as global variables. This section must be declared outside of all the functions.

Function main

Every program written in C language must contain main () function. The function main() is a starting point of every C program. The execution of the program always begins with the function main ().

Declaration Part

The declaration part declares the entire variables that are used in executable part. The initializations of variables are also done in this section. Initialization means providing initial value to the variables

Executable Part

This part contains the statements following the declaration of the variables. This part contains a set of statements or a single statement. These statements are enclosed between the braces.

User Defined Function

The functions defined by the user are called user-defined functions. These functions are generally defined after the main () function.

3. Algorithm and flow chart

i) Max & Min of 2 No.s

ii) Max & Min of 3 No.s

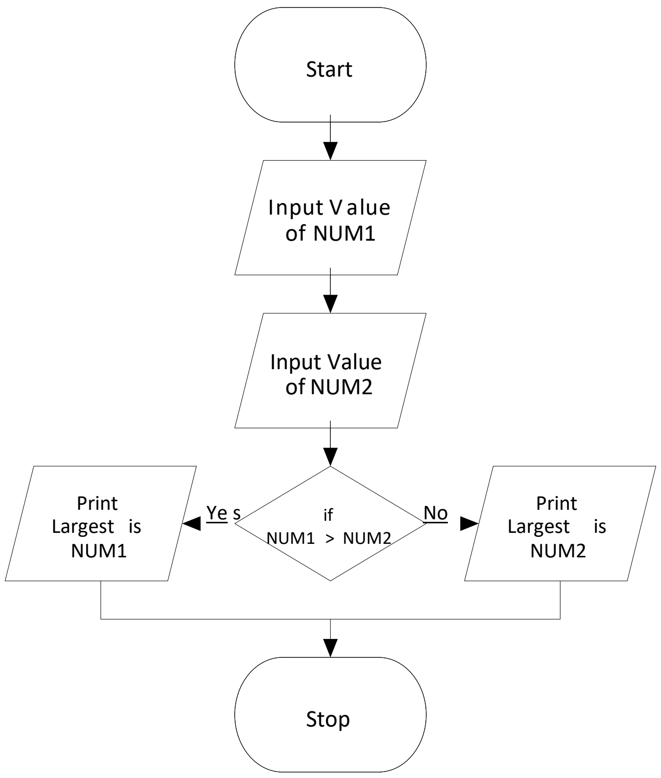
iii) To check whether a no. is prime or not.

iv) Factorial of a number

Ans:

(i) Algorithm & Flowchart to find the larger of two numbers

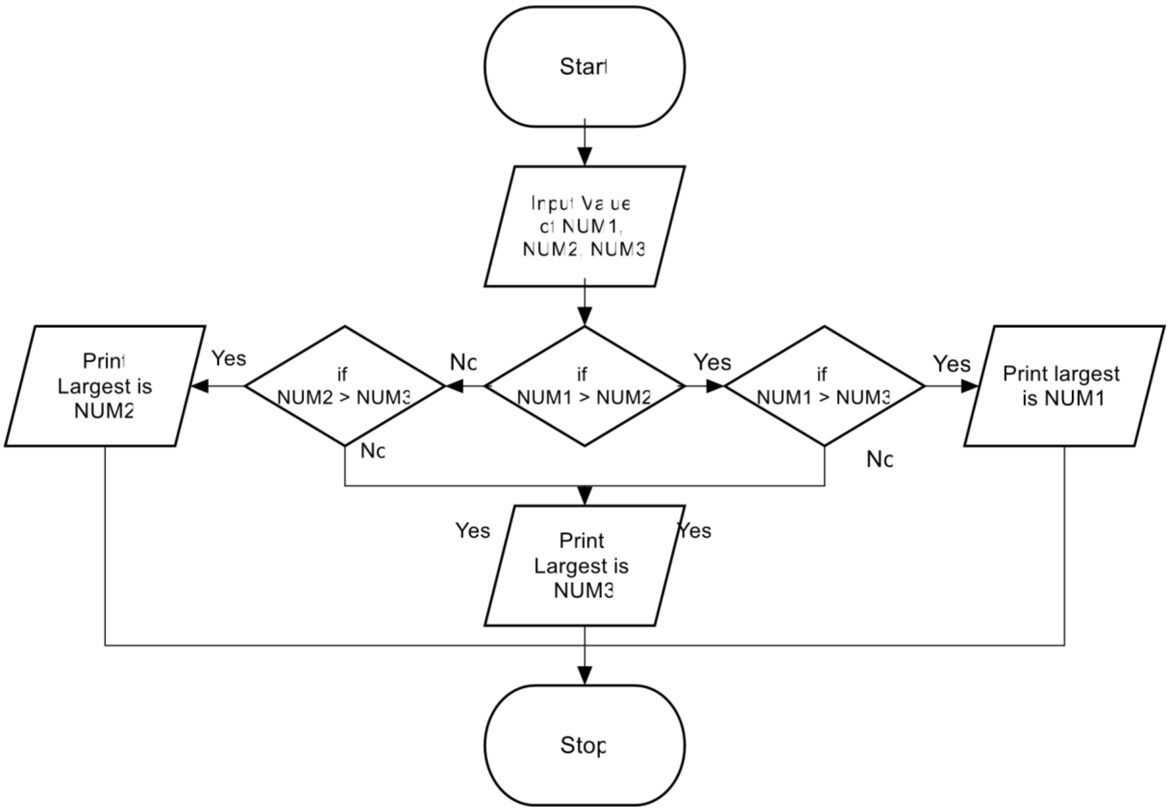
```
Step-1   Start
Step-2   Input two numbers say
NUM1, NUM2
Step-3    IF NUM1 > NUM2 THEN
                Print largest is NUM1

ELSE              print largest is NUM2

          ENDIF
Step-4    Stop
```



(ii).Find the largest of three numbers

Algorithm

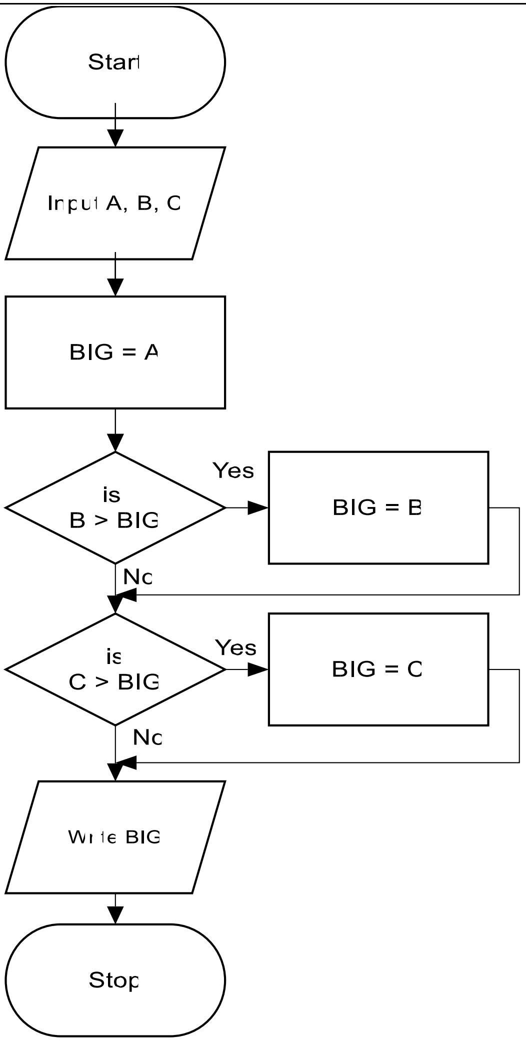
```
Step-1   Start
Step-2   Read three numbers say num1, num2, num3
Step-3    if num1>num2 then go to step-5
Step-4    IF num2>num3 THEN
              print num2 is largest
ELSE
              print num3 is largest
             ENDIF
              GO TO Step-6
 Step-5   IF num1>num3 THEN
               print num1 is largest
ELSE
              print num3 is largest
            ENDIF
Step-6   Stop
```



Algorithm & Flowchart to find the largest of three numbers (an another way)

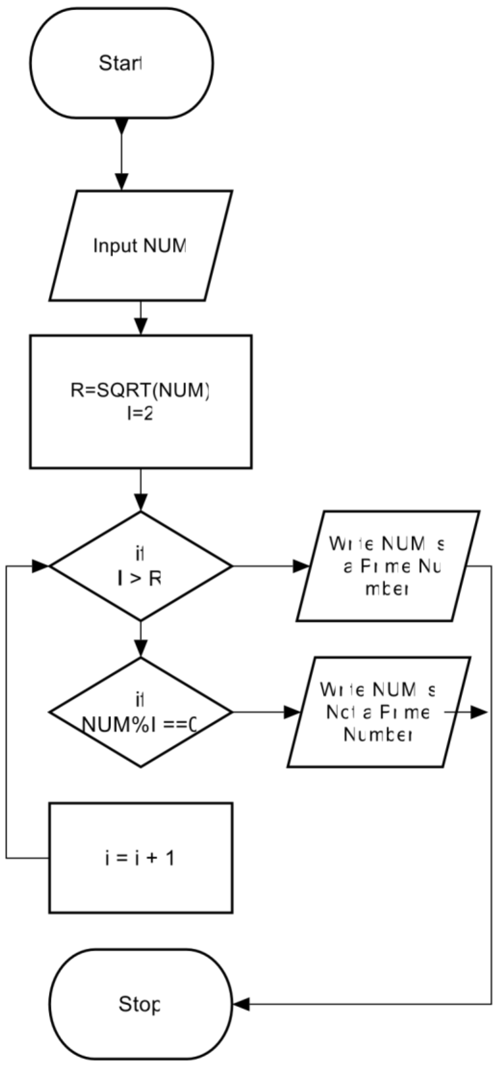
Algorithm

```
Step-1   Start
Step-2   Read three numbers say A,B,C
Step-3   BIG = A
Step-4   IF B > BIG THEN
             BIG = B
         ENDIF
Step-5   IF C >BIG  THEN
             BIG = C
         ENDIF
Step-6   Write BIG
Step-7   Stop
```



(iii).To check if a number is prime or not.

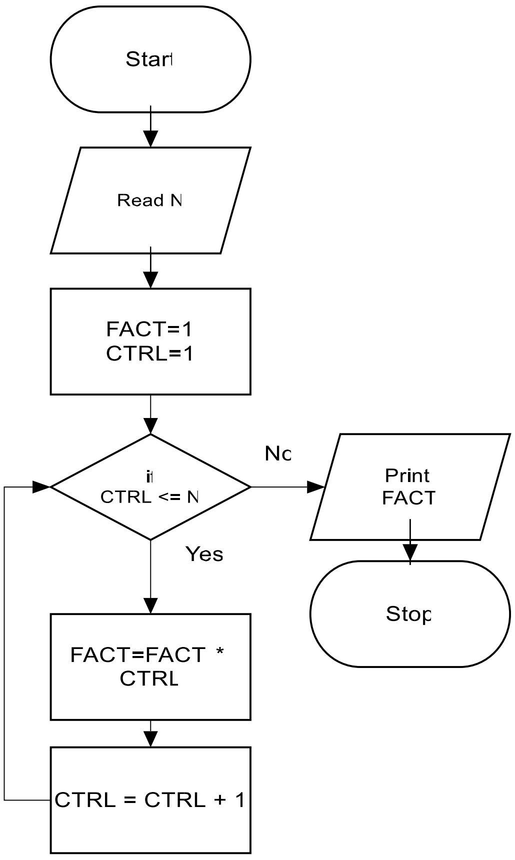
```
Step-2    Input NUM
Step-3    R=SQRT(NUM)
Step-4    I=2
Step-5    IF ( I > R) THEN
              Write NUM is Prime Number
              Stop
          ENDIF
Step 6    IF ( NUM % I ==0) THEN
              Write NUM is Not Prime
              Stop
          ENDIF
Step-7    I = I + 1
Step-8    Go to Step-5
```



(iv) . To find factorial of a number.

Algorithm & Flowchart to find Factorial of number n (n!=1x2x3x…n)

```
Algorithm
Step-1   Start
Step-2   Read number N
Step-3   FACT=1  CTRL=1
Step-4   WHILE (CTRL <= N)
           DO
             FACT=FACT*I
             CTRL=CTRL+1
         DONE
Step-5   Display FACT
Step-6   Stop
```



4. Operators used in C language.

Ans:

OPERATORS IN C:

1. ARITHMETIC OPERATORS:

Operator	*Meaning*
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division (gives Quetient)
%	Modulo Division (gives Remainder)

2. RELATIONAL OPERATORS:

Executable C statements either performs actions (such as calculations or input or output of data) or make decisions. We might make a decision in a program, for example, to determine if a person is passed or failed in an exam. To check the condition(for example in if control structure) we use Relational operators.

Operator		*Meaning*
>	x > y	x is greater than y
<	x < y	x is less than y
>=	x>=y	x is greater than or equal to y
<=	x<=y	x is less than or equal to y
==	x == y	x is equal to y
!=	x != y	x is not equal to y

4. <u>LOGICAL OPERATORS:</u>

C provides logical operators that may be used to form more complex condition by combining simple conditions. The logical operators are

Operator	*Meaning*
&&	logical AND
||	logical OR
!	logical NOT

Like the simple relational expressions(x>y), a logical expression also yields a value of zero or one.

AND operator:

(cond1&&cond2)

If both conditions are true the output of the logical AND expression is true(i.e it returns 1) and if any of the condition is false the output is false(i.e. it returns 0)

OR Operator:

(condt1 || condt2)

If any of the condition is true the output of the expression is true and the output will be false when both the conditions are false

Not Operator:

(!conditon)

If the condition is true the output of the expression will be false and if the condition is false the output of the expression is true.

5. <u>ASSIGNMENT OPERATORS:</u>

These ops. are used to assign the result of an expression to a variable . In addition to the usual assignment op. '=' , C has a set of shorthand assignment ops. of the form.

v op=exp;

The operator op= is known as the shorthand operator. The above statement can be written as

v=v op (exp);

Ex: x=8; y=5;

x+=y+1; (here += is shorthand op. Similarly we can also use -=, *=, /=, %= as shorthand ops.)

The above statement is same as

x=x + (y+1); (here x value becomes 14)

6. INCREMENT AND DECREMENT OPERATORS:

C has two very useful operators called increment and decrement operators. The operator ++ add 1 and operator -- subtracts 1. Both are unary operators (because we are operating on one operand).

preincrement	++m	equivalent to
postincrement	m++	m=m+1; or m+=1;
predecrement	--m	equivalent to
postdecrement	m--	m=m-1; or m+=1;

we use increment and decrement statements in for and while loops extensively.

When m++ and ++m mean the same thing when they form statements independently, they behave differently when they are used in expressions on the right-hand side of an assignment statement.

7. CONDITIONAL OPERATOR:(Ternary Operator)

The C language has an unusual operator, useful for making two-way decisions. This operator is a combination of ? and : and takes three operands(therefore called as ternary operator). This operator is popularly known as the conditional operator. The general form of use of the conditional operator is as follows:

conditional expression ?

The conditional expression is evaluated first. If the result is nonzero(true) expression1 is evaluated and returned, otherwise expression 2 is evaluated and its value is returned.

8. BITWISE OPERATORS:

These ops. are used for manipulation of data at bit level. These ops. are used for testing the bits or shifting them right or left.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
~	One’s complement
<<	Shift left
>>	Shift right

(UNIT-2)

Short Answer Questions:

1. Differentiate between while and do-while loop.

Ans:

*While Loop:

A **while** loop in C programming repeatedly executes a set of statements or statement as long as a given condition is true. while statement consists of while followed by controlling expression enclosed within parentheses.

Syntax:

The syntax of a **while** loop in C programming language is:

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

*Do-While Loop:

Unlike **for** and **while** loops, which test the loop condition at the top of the loop, the **do-while** loop in C programming checks its condition at the bottom of the loop.

A **do-while** loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

The do-while statement consists of a do keyword followed by statement or statements that constitute do-while body, followed by the while clause consisting of while keyword followed by do-while controlling expression enclosed with in parenthesis. The while clause is terminated with a semicolon.

Syntax

The syntax of a **do-while** loop in C programming language is:

```
do
 {
Statement(S);
}
While(condition);
```

2. Conditional / ternary Operators.

Ans:

*C Ternary Operator (?:)

A conditional operator is a ternary operator, that is, it works on 3 operands.

Conditional Operator Syntax

```
conditionalExpression ?expression1 : expression2
```

The conditional operator works as follows:

- The first expression conditionalExpression is evaluated at first. This expression evaluates to 1 if it's and evaluates to 0 if it's false.
- If conditionalExpression is true, expression1 is evaluated.
- If conditionalExpression is false, expression2 is evaluated.

Example : C conditional Operator

```
#include<stdio.h>
Void main()
Char February;
Int days;
printf("If this year is leap year, enter 1. If not enter any integer: ");
scanf("%c",&February);

// If test condition (February == 'l') is true, days equal to
29. // If test condition (February =='l') is false, days equal
to 28.
days=(February=='1')?29:28;

     printf("Number of days in February = %d",days);
     return0; }
```

Output:

```
If this year is leap year, enter 1. If not enter any integer: 1
Number of days in February = 29
```

3. Bitwise Operators.

Ans:

*BITWISE OPERATORS:

These operators are used for manipulation of data at bit level. These operators are used for testing the bits or shifting them right or left.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
~	One's complement
<<	Shift left
>>	Shift right

4. Differentiate between break, continue, goto and exit statements.

Ans:

*Break Statement:

The **break** statement in C programming has the following two usages:

When a **break** statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop. It can be used to terminate a case in the **switch**statement .

If you are using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax

The syntax for a **break** statement in C is as follows:

```
break;
```

*Continue Statement**:**

The **continue** statement in C programming works somewhat like the **break** statement. Instead of forcing termination of loop, it bypasses the current iteration of the loop to take place, skipping any code in between.

For the **for** loop, **continue** statement causes the conditional test and increment portions of the loop to execute. For the **while** and **do...while** loops, **continue** statement causes the program control to pass to the conditional tests.

Syntax

The syntax for a **continue** statement in C is as follows:
continue;

Goto Statement:

A **goto** statement in C programming provides an unconditional jump from the 'goto' to a labelled statement in the same function.

Use of **goto**statement is highly discouraged in any programminglanguage because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten to avoid them.

Syntax

The syntax for a **goto** statement in C is as
follows: goto label;

```
.
.

.
label: statement;
```

Here **label** can be any plain text except C keyword and it can be set anywhere in the C program above or below the **goto** statement.

Exit Statement:

exit() is a standard library function.

exit() terminates program execution when it is called.

exit() can be used as a variable name.

stdlib.h needs to be included in order to use exit().

exit() returns the control to the operating system or another program that uses this one as a sub-process.

Example of exit()

```
while(true){
….
  if(condition)
    exit(-1);
}
```

Long Answer Questions:

1. Switch Statement with example.

Ans:

The switch selection structure:

The switch selection structure performs one of many different actions depending on the value of an expression. The switch structure is called multiple-selection structure because it selects among many different actions.

```
switch (expression)
{
      case const1:
                  block-1;
                  break;
      case const2:
                  block-2;
                  break;
      ……….
      ……….
      default :
                default block;
                break;
}
statement-x;
```

Example:

```
/* Calculator program using switch case  */
#include<stdio.h>
#include<conio.h>
 void main()
{
  int a,b,c,n;
  clrscr();
  printf(“Enter two no’s\n”);
  scanf(“%d %d”, &a,&b);
  printf(“1.Addition\n2.Substration\n3.Multiplication\n4.Division
  scanf(“%d”,&n)
  switch(n)
  {
          case 1:c=a+b;
               printf(“%d”,c);
               break;
          case 2:c=a-b;
               printf(“%d”,c);
         break;
          case 3:c=a*b;
               printf(“%d”,c);
               break;
          case 4:c=a/b;
               printf(“%d”,c);
               break;
          default: printf(“Invalid Number\n”);
                break;
   }
 getch();
  }
```

2. Programs:

   i) To check whether a given no is palindrome or not.
   ii) Sin (x) and Cos(x) using series expansion.

Ans:

(i) To check whether a given no is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num,rev=0,rem,temp;
clrscr();
printf("enter a number:\n");
scanf("%d",&num);
temp=num;
while(num>0)
{
rem=num%10;
num=num/10;
rev=rev*10+temp;
}
if(rev==temp)
printf("it is a palindrome");
else
printf("it is not a palindrome");
getch();
}
```

Output:

```
enter a number :
101
It is a palindrome
```

(ii). Sin(x) using sine series:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int degree,i=1;
double radian,term,nr,dr,sum=0.0;
clrscr();
printf("Enter a value in degree:\n");
scanf("%d",&degree);
radian=degree*0.01745329;
nr=radian;
dr=1.0;
term=nr/dr;
while(i<=20)
{
sum=sum+term;
nr=(-nr)*radian*radian;
dr=dr*(i+1)*(i+2);
term=nr/dr;
i=i+2;
}
printf("sine value =%lf",sum);
getch();
}
```

Output:

```
Enter a value in degrees:
90
Sine value=1.000000
```

(iii). Cos(x) using cos series expansion:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int degree,i=2;
double radian,term,nr,dr,sum=1.0;
clrscr();
printf("Enter a value in degree:\n");
scanf("%d",&degree);
radian=degree*0.01745329;
nr= -radian*radian;
dr=2.0;
term=nr/dr;
while(i<=20)
{
sum=sum+term;
nr=(-nr)*radian*radian;
dr=dr*(i+1)*(i+2);
term=nr/dr;
i=i+2;
}
printf("cosine value =%lf",sum);
getch();
}
```

Output:

```
Enter a value in degrees:
90
Cosine value=0.00000
```

4. Program to find roots of quadratic equation.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
float root1,root2,a,b,c;
int select:
printf("enter the coefficients\n");
scanf("%f%f%f",&a,&b,&c);
term=b*b-4*a*c;
temp=2*a;
if(term==0)
select=0;
elseif(term>0)
select=1;
else
select=2;
switch(select)
{
case 0: root1=-1*b/temp;
root2=root1;
printf("The roots are real and unequal:\n");
printf("root1=%f and root2=%f",root1,root2);
break;
case 1: term=sqrt(term);
root1=(-1*b+term)/temp;
root2=(-1*b-term)/temp;
printf("the roots are real and unequal:\n");
printf("root1=%f and root2=%f",root1,root2);
break;
case2:printf("the roots are imaginary:\n");
break;
}
getch();
}
```

3. Searching Techniques –

* linear
* Binary

Ans:

*Algorithm for Linear search:

```
Step 1: Start
Step 2: Position= -1
Step 3: Read array size n ,elements of array  and key
Step 4: i=0
Step 5: if(a[i]==key) then position=i
Step 6: i++
Step 7: if(i<n) then go to Step 5
Step 8: if(position= -1) then print element not found
Step 9: else
             print element found at position +1 location
Step 10: Stop.
```

*Algorithm for Binary Search:

```
Step 1: Start
Step 2: Position= -1
Step 3: Read array size n ,elements of array  and key
Step 4: L=0 and H=n
Step 5: Mid=low+high/2
Step 6: if a[mid]==key then position==middle
Step 7: else if
             if key<a[mid] then
             high=mid-1
Step 8: if key>a[mid] then
            low=mid+1
Step 9: if(low<high) goto Step 5
Step 10: if(position=1) then print not found
              else
              print element at middle
Step 11: Stop
```

6.Sorting Techniques-　　　* Selection Sort

　　　　　　　　　　　　　* Bubble Sort

*Selection Sort:

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparision based algorithm in which the list is divided into two parts the sorted part at the left end and the unsorted part at the right end. Initially the sorted part is empty and the unsorted part is the entire list.
If the user wants to sort an array in ascending order then the comparision is made between two elements and the smaller elements is placed at the first place. The process is repeated untill last elements are compared.

EX:

2	150	113	19	225	50	71

*Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int A[100];
int small,N,i,j,temp;
clrscr();
printf("enter the size of array:\n");
scanf("%d",&N);
printf("Enter array elements:\n",N);
for(i=0;i<n;i++)
{
small=i;
for(j=i+1;j<N;j++)
{
if(A[j]<A[small])
{
small=j;
}
}
temp=A[i];
A[i]=A[small];
A[small]=temp;
}
printf("The sorted array list is :\n");
for(i=0;i<n;i++)
printf("%d\t",A[i]);
getch();
}
```

Output:

```
Enter the size of array:
5
Enter array elements:
21
45
67
2
6
The sorted array list is:
2   6  21  45  67
```

Bubble Sort:

1. An array with N elements is taken.
2. Compare the first two elements in the arrary a1 and a2. If a2<a1 then interchange their values.
3. Compare a2 and a3 interchange them if a3<a2
4. Continue the process till the last time elements are compared and interchanged.
5. Repeat the above process (N-1) times.
6. In repeated trips through the array largest element is bubbled upto the top thus the name bubble sort.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100];
int trip,N,i,j,temp;
clrscr();
printf(“enter the size of array:\n”);
scanf(“%d”,&N);
printf(“Enter array elements:\n”,N);
for(i=0;i<n;i++)
{
scanf(“%d”,&a[i]);
}
for(trip=1;trip<N;trip++)
{
for(i=0;i<N-trip;i++)
{
if(a[i]>a[i+1])
{
temp=a[i];
a[i]=a[i+1];
a[i+1]=temp;
}
}
}
printf(“The sorted array list is :\n”);
for(i=0;i<N;i++)
printf(“%d\t”,a[i]);
getch();
}
```

Output:

```
Enter the size of array:
5
Enter array elements:
21
45
67
2
6
The sorted array list is:
2   6  21  45  67
```

7. To print array elements in reverse order.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
void reverse(int*,int)
int a[10],n,i;
clrscr();
printf("enter the size:\n");
scanf("%d",&n);
printf("enter %d array elements",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
reverse(a,n);
getch();
}
void reverse(int*a,int n)
{
int i;
for(i=n-1;i>=0;i--)
printf("%d\t",a[i]);
}
```

Output:

```
enter the size:
5
Enter 5 array elements:
12
46
3
9
8
8   9   3   46   12
```

(UNIT-3)

Short Answer Questions:

1 Read and display elements of a multi dimensional array.

Ans:

Multi dimensional array

Multi dimensional array can have three, four or more dimensions. The first dimension is called plane which consists of rows and columns. The three dimensional array to be an array of two dimensional arrays. It considers the two dimensional array to be an array of one dimensional arrays.

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

```
int test[2][3][4] = {
                    {//plane 0
                      {3, 4, 2, 3}, //row 0
                      {0, -3, 9, 11}, //row 1
                      {23, 12, 23, 2} //row 2
                    },
                    {//plane 1
                      {13, 4, 56, 3}, //row 0
                      {5, 9, 3, 5},  // row1
                      {3, 1, 4, 9} //row 2
                    }
                       };
```

2. Function Prototype.

Ans:

Function Prototype

A function prototype tells the compiler the number and type of arguments that are to passed to function and the type of value that is to be returned by the function. The void is used if no value is returned by the function.

Example : int sum(int,int);

3. Difference between function declaration & function definition.

Ans:

Difference between Declaration and Definition

S.No.	Declaration	Definition
1	Tells compiler about name and type of variable, class, function, etc.	Tells compiler about what value stored in variable or working of function, class, etc.
2	Memory allocation is not done.	Memory allocation is done.
3	Can re-declare multiple times.	Can define only once.

4. Syntax of defining a function.

Ans:

<u>*Syntax:</u>

```
return-type function-name(argument decleration)
{
local declerations
………
………
```

<u>Long Answer Questions:</u>

1 String handling functions-{ strlen(), strcat(), strcmp(), strcat(), strrev()-etc} with and without using library function.

Ans:

String handling functions c provides in-built functions to perform different operations on strings. To use string functions , the header file <string.h> is to be included.

1) strcat(string1, string2); char string1[35]=”Yahoo”,string2[20]=”Hotmail”;

   strcat(string1,string2);

   strcat(string1,”Messanger”);

2) strcpy(string1, string2); char string1[35],string2[20]=”Hotmail”;

   strcpy(string1,string2);

   strcpy(string1,”Yahoo”);

3) strlen(string1); char string1[35]=”CVR College of Engineering”;int n; n= strlen(string1);

4) strcmp(string1, string2);

   char string1[35]=”Yahoo”,string2[20]=”Hotmail”;

   the strcmp function returns zero if strin1=string2, returns negative value if strin1<string2 or positive value if strin1>string2

4. To check whether a given string is palindrome or not.

Ans:

```
#include <stdio.h>

#include <string.h>

 int main()

{

  char a[100], b[100];

   printf("Enter a string to check if it is a palindrome\n");

  gets(a);

 strcpy(b, a);  // Copying input string

  strrev(b);  // Reversing the string
```

```
   if (strcmp(a, b) == 0)  // Comparing input string with the reverse string

    printf("The string is a palindrome.\n");

  else

    printf("The string isn't a palindrome.\n");

return 0;

      }
```

Output:

```
Enter a string to check if it is a palindrome
wow
The string is a palindrome.
```

3.To arrange a given string in alphabetical order.
Ans:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

 int main()

{

 char ch, input[100], output[100];

  int no[26] = {0}, n, c, t, x;

   printf("Enter some text\n");

   scanf("%s", input);

  n = strlen(input);

 for (c = 0; c < n; c++)

  {

    ch = input[c] - 'a';
    no[ch]++;
 }
  t = 0;
  for (ch = 'a'; ch <= 'z'; ch++)
  {
   x = ch - 'a';
    for (c = 0; c < no[x]; c++)
   {
     output[t] = ch;
     t++;
   }
  }
  output[t] = '\0';
  printf("%s\n", output);
  return 0;
}
```

Output:

```
Enter some text
game
aegm
```

(UNIT-4)

Short Answer Questions:

1 Define recursion with an example.

Ans:

A function that calls itself is called as recurssive function and this technique is known as recurssion. Recurssion continues untill some condition is net to prevent it.

*Ex:

A function to evaluate factorial of n is as follows.

```
factorial(int n)

{

int fact;

if(n==1)

return(1);

else

fact=n*factorial(n-1)

return(fact);

}
```

2. Define a structure with an example.

Ans:

Structure Definition

A structure is a collection of logically related elements, possibly of different types, having a single name.

*Syntax:

```
structure type name

{

type1 member1;

type2 member2;

};
```

4. Operations performed on structures.

Ans:

The following operations can be performed on structures:

1. Traversing- It is used to access each data item exactly once so that it can be processed.

2. Searching- It is used to find out the location of the data item if it exists in the given collection of data items.

3. Inserting- It is used to add a new data item in the given collection of data items.

4. Deleting- It is used to delete an existing data item from the given collection of data items.

5. Sorting- It is used to arrange the data items in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

6. Merging- It is used to combine the data items of two sorted files into single file in the sorted form.

Long Answer Questions:

1 write a recursive function to
   i) Find factorial
   ii) GCD
   iii) Primary search using recursion
   iv) Fabonacci series
   v) X^n

Ans:

i) Find Factorial

```
#include <stdio.h>

#include<conio.h>

int fact(int);

void main()

{

int num,f;

clrscr();

printf(“Enter a number:\n”);

scanf(“%d”,&num);

f=fact(num);

printf(“factorial of %d is: %d ,num,f);

getch();

}

int fact(int n)

{

if(n==1)

return 1;

else

return(n*fact(n-1));

}
```

*Output:

```
Enter a number:

5

Factorial of is 120
```

ii)GCD of two numbers

```
 #include <stdio.h>
int hcf(int n1, int n2);
int main()
{
   int n1, n2;
   printf("Enter two positive integers: ");
   scanf("%d %d", &n1, &n2);

   printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1,n2));
   return 0;
}

int hcf(int n1, int n2)
{
    if (n2 != 0)
       return hcf(n2, n1%n2);
    else
       return n1;
}
```

Output:

```
Enter two positive integers: 366
60
G.C.D of 366 and 60 is 6.
```

iv) Fibonacci series:

Finding Fibonacci series of a number using recursion *

```
#include<include.h>

#include<conio.h>

Void printFibonacci(int); void main()

{ int,n; long int i=0,j=1,f;

clrscr(); printf("Enter the

range of the Fibonacci

series:"); scanf("%d",&n);
printf(“Fibonacci Series:

“); printf(“%d %d”, i,j);

printFibonacci(n);

getch();

}

Void  printFibonacci(int n)

{ static long int first=0,second=1,sum; if(n>2) { sum =

first + second; first = second; second = sum; printf(“%ld

“,sum); printFibonacci(n-1);

}

}
```

2. Difference between Arrays and Structures.

BASIS FOR COMPARISON	ARRAY	STRUCTURE
Basic	An array is a collection of variables of same data type.	A structure is a collection of variables of different data type.

Syntax	type array_name[size];	struct sruct_name{
		type element1;
		type element1;
		.
		.
		} variable1, variable2, . .;

Memory	Array elements are stored in contiguous memory location.	Structure elements may not be stored in a contiguous memory location.
Access	Array elements are accessed by their index number.	Structure elements are accessed by their names.
Operator	Array declaration and element accessing operator is "[ ]" (square bracket).	Structure element accessing operator is "." (Dot operator).

BASIS FOR COMPARISON	ARRAY	STRUCTURE
Pointer	Array name points to the first element in that array so, array name is a pointer.	Structure name does not point to the first element in that structure so, structure name is not a pointer.
Objects	Objects (instances) of an array can not be created.	Structure objects (instance or structure variable) can be created.
Size	Every element in array is of same size.	Every element in a structure is of different data type.
Bit filed	Bit filed can not be defined in an array.	Bit field can be defined in a structure.
Keyword	There is no keyword to declare an array.	"struct" is a keyword used to declare the structure.
User-defined	Arrays are not user-defined they are directly declared.	Structure is a user-defined datatype.
Accessing	Accessing array element requires less time.	Accessing a structure elements require comparatively more time.
Searching	Searching an array element takes less time.	Searching a structure element takes comparatively more time than an array element.

3. Program illustrating self referntial structure.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct detail
}
char name[20];
int age;
struct detail*ptr;
};
struct detail p1={“john”,60};
struct detail p2={“smith”,61}
struct detail p3={“mary”,45}
p1.ptr=&p2;
p2.ptr=&p3;
p3.ptr=0;
clrscr();
printf(“name 1: %s\t Age 1: %d\n”,p1.name,p1.age);
printf(“name 2:%s\t Age 2: %d\n”,p2.name,p2.age);
printf(“name 3 :%s\t Age 3: %d\n”,p3.name,p3.age);
getch();
}
```

(UNIT-5)

Short Answer Questions:

1.Define a Pointer variable.

Ans:

Declaring a pointer variable

In C , every variable must be declared before they are used. Since the pointer variables contain address that belongs to a separate data type, they must be declared as pointers before we use them.

The syntax for declaring a pointer variable is as follows,

```
data type *ptr-name;
```

2.Define a file in C language.

Ans:

FILES:

A file is an external collection of related data treated as a unit. The primary purpose of a file is to keep record of data. Record is a group of related fields. Field is a group of characters they convey meaning.

Creating a Stream:

We can create a stream when we declare it. The declaration uses the FILE type as shown below,

FILE*fp;

3.Syntax for opening a file.

Ans:

*Opening syntax of a file:

FILE*fp;

fp=fopen(“file name”,”mode”);

4.Error handling functions in files.

Ans:

C language does not provide any direct support for error handling. However a few methods and variables defined in error.h header file can be used to point out error using the return statement in a function. In C language, a function returns -1 or NULL value in case of any error and a global variable errno is set with the error code. So the return value can be used to check error while programming.

5.Types of files in c language.

Ans: There are 4 types of files in c language:

1.Source Code File

This file includes the source code of the program.

The extension for these kind of files are '.c'. It defines the main and many more functions written in C.

main() is the starting point of the program. It may also contain other source code files.

2.Header Files

They have an extension '.h'. They contain the C function declarations and macro definitions that are shared between various source files.

Common standard header files are:

i) string.h – used for handling string functions.

ii) stdio.h – used for giving standardized input and output.

iii) math.h – used for mathematical functions.

iv) conio.h – used for clearing the screen.

3.Object files

They are the files that are generated by the compiler as the source code file is processed.

These files generally contain the binary code of the function definitions.

The object file is used by the linker for producing an executable file for combining the object files together. It has a '.o' extension.

4.Executable file

This file is generated by the linker.

Various object files are linked by the linker for producing a binary file which will be executed directly.

They have an '.exe' extension.

6.What are input & output Streams.

Ans:

STREAMS

1. A stream is a general name given to a flow of data.
2. All input and output is performed with streams.
3. A "stream" is a sequence of characters organized into lines.
4. Each line consists of zero or more characters and ends with the "newline" character.

7.Various arithmetic operations performed on pointers.

Ans:

We can perform addition and subtraction on pointers.

consider int s,*px; px=&x; x=25;

Assume the memory address of x is 5003, if px=px+3; is executed then the px will contain 5009 is 5003+3*(size of integer) i.e. 5003+3*2=5009

i.e. base address + number * size of datatype

++*px; // increments the value of variable x by 1

--*px // decrement x by 1

8.Modes of a file.

Ans:

File Mode:

When we open a file, we explicitly define its mode. The mode shows how we will use the file: for reading, for writing, or for appending.

1. “r” (read) mode:

The read mode (r) opens an existing file for readingIf we try to write a file opened in read mode, we get an error message.

Syntax fp=fopen (“filename”,”r”);

2. “w” (write) mode

The write mode (w) opens for writing. If the file doesn’t exist, it is created.

Syntax fp=fopen (“filename”,”w”);

3. “a” (append) mode

The append mode (a) also opens an existing for writing. Instead of creating a new file, however, the writing starts after the last character; that is new data is added, or appended, at the end of the file.

IF the file doesn’t exist, it is created and opened.

Syntax fp=fopen (“filename”,”a”);

4. “r+” (read and write) mode

In this mode file is opened for both reading and writing the data. If a file does not exist then NULL, is returned.

Syntax: fp=fopen (“filename”,”r+”);

5.“w+” (read and write) mode

In this mode file is opened for both writing and reading the data. If a file already exists its contents erased. If a file does not exist then new file created.

Syntax: fp=fopen (“filename”,”w+”);

6.“a+” (append and read) mode

In this mode file is opened for reading the data as well as data can be added at the end.

Syntax: fp=fopen (“filename”, “a+”);

Long Answer Questions:

1. Write about paramter passing functon with the help of example.

Ans:

Passing parameters to functions:

1. Call by Value / Pass by Value
2. Call by Reference / Pass by Reference

Call by Value:

When the value is passed directly to the function it is called call by value.

In call by value only a copy of the variable is passed so any changes made to the variable does not reflects in the calling function.

EX:

```
#include<stdio.h>
#incude<conio.h>
swap(int,int);
void main()
{
int x,y;
printf("enter two no's\n");
scanf ("%d%d",&x,&y);
printf("before swapping x=%d,y=%d",x,y);
swap(x,y)
getch();
}
swap(int a,int b)
{
int t;
t=a;
a=b;
b=t;
printf("after swapping: x=%d,y=%d,a,b);
}
```

Output:

```
Enter two no's: 7  21
Before swapping: 7  21
After swapping: 21  7
```

*Call by Reference:

When the address of the value is passed to the function is called call by reference.

In call by reference since the address of the value is passed any changes made to the value effects in the calling function.

*EX:

```
#include<stdio.h>
#incude<conio.h>
swap(int*,int*);
void main()
{
int x,y;
printf("enter two no's\n");
scanf ("%d%d",&x,&y);
printf("before swapping x=%d,y=%d",x,y);
swap(&x,&y)
printf("after swapping: x=%d,y=%d",x,y);
getch();
}
swap(int *a,int *b)
{
int t;
t=*a;
a*=b*;
b*=t;
}
```

*Output:

```
Enter two no's: 10  11
Before swapping: 10  11
After swapping: 11  10
```

2 Dynamic memory allocation

Ans:

The process of allocating memory during program execution is called dynamic memory allocation.

Dynamic Memory Allocation functions in c

C language offers 4 dynamic memory allocation functions. They are,

1. malloc()
2. calloc()
3. realloc()
4. free()

<u>1.malloc function:</u>

- malloc () function is used to allocate space in memory during the execution of the program.
- malloc () does not initialize the memory allocated during execution. It carries garbage value.
- malloc () function returns null pointer if it couldn’t able to allocate requested amount of memory.

Syntax of malloc()

ptr = (cast-type*) malloc(byte-size)

<u>2. calloc function:</u>

• calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn’t.

The name calloc stands for "contiguous allocation".

The only difference between malloc() and calloc() is that, malloc() allocates single block of memory whereas calloc() allocates multiple blocks of memory each of same size and sets all bytes to zero. Syntax of calloc()

ptr = (cast-type*)calloc(n, element-size);

<u>3.realloc function:</u>

realloc () function modifies the allocated memory size by malloc () and calloc () functions to new size. If enough space doesn’t exist in memory of current block to extend, new block is allocated for the full size of reallocation, then copies the existing data to new block and then frees the old block.

<u>4. free function:</u>

free () function frees the allocated memory by malloc (), calloc (), realloc () functions and returns the memory to the system free()

Dynamically allocated memory created with either calloc() or malloc() doesn't get freed on its own. You must explicitly use free() to release the space. syntax of free() free(ptr);

This statement frees the space allocated in the memory pointed by ptr.

Example: Using C malloc() and free()

3 File handling functions

Ans:

File handling functions	Description
fopen ()	fopen () function creates a new file or opens an existing file.
fclose ()	fclose () function closes an opened file.
fgetc ()	fgetc () function reads a character from file.
fputc ()	fputc () functions write a character to file.
gets ()	gets () function reads line from keyboard.
puts ()	puts () function writes line to o/p screen.
fgets ()	fgets () function reads string from a file, one line at a time.
fputs ()	fputs () function writes string to a file.
feof ()	feof () function finds end of file.
fgetchar ()	fgetchar () function reads a character from keyboard.
fprintf ()	fprintf () function writes formatted data to a file.
fscanf ()	fscanf () function reads formatted data from a file.
fputchar ()	fputchar () function writes a character onto the output screen from keyboard input.
getc ()	getc () function reads character from file.
getch ()	getch () function reads character from keyboard.
getchar ()	getchar () function reads character from keyboard.

putc ()	putc () function writes a character to file.
putchar ()	putchar () function writes a character to screen.
printf ()	printf () function writes formatted data to screen.
scanf ()	scanf () function reads formatted data from keyboard.

4 To count no. of characters, words and files in a file.
5 Copy contents of one file into another file.
6 Compare the contents of two files.
7 Merging of 2 files into another file.

Refer Lab Manual For Programs