Questions and Answers

FOR

Programming for Problem Solving

I YEAR I SEMESTER B.TECH

(COMPUTER SCIENCE AND ENGINEERING)

2020-21

Department of Computer Science and Engineering

ACE ENGINEERING COLLEGE

(NBA Accredited B.Tech for All Courses with NAAC 'A' Grade)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)

Ankushapur (V), Ghatkesar (M), R.R.Dist - 501 301.

UNIT 1:

1. Explain in detail the structure of C Program and procedure to create, compile and run the C program.

Answer:

Structure of C program

• Structure of C program is defined by set of rules called protocol, to be followed by programmer while writing C program.

• All C programs are having sections/parts which are mentioned below.

S.No	Sections	Description
1	Documentation section	We can give comments about the program, creation or modified date, author name etc in this section Single line comment - example : //sample program Multiline comment - Example : /* comment line1 comment line2 comment3 */
2	Link Section	Header files that are required to execute a C program are included in this section
3	Definition Section	In this section, variables are defined and values are set to these variables.
4	Global declaration section	Global variables are defined in this section. When a variable is to be used throughout the program, can be defined in this section.
5	Function Prototype	Function prototype gives many information about a declaration section function like return type, parameter names used
6	Main function	Every C program is started from main function and this function contains two major sections called declaration section and executable section.
7	User defined function section	User can define their own functions in this section which perform particular task as per the user requirement.

Creating and Running Programs:

Computer hardware understands a program only if it is coded in its machine language. It is the job of the programmer to write and test the program.

There are four steps in this process:

1. **Writing and Editing the program**
   - The software used to write programs is known as a text editor. A text editor helps user to enter, change, and store character data
2. **Compiling the program**
   - The code in a source file stored on the disk must be translated into machine language;
   - this is the job of the compiler.
   - The 'c' compiler is two separate programs. The preprocessor and the translator.
   - The code generated after compilation is called object code.
   - The preprocessor reads the source code and prepares it for the translator.
   - While preparing the code, it scans for special instructions known as preprocessor commands.
   - After the preprocessor has prepared the code for compilation, the translator convert the program into machine language and generate the object code that is, not executable because it does not have the required C and other functions included

3. **Linking the program with the required library modules**
    - A C program is made up of many functions. Function can be user defined or predefined
    - Predefined function, such as input/output function and mathematical library functions that exist elsewhere and must be attached to our program.
    - The linker assembles all of these functions code to object code and then generate final executable program.
4. **Executing the program.**
    - Once program has been linked, it is ready for execution.
    - To execute a program we use an operating system command, such as run
    - OS use the program called loader to load program from secondary memory into primary memory to execute the program

2. What is Operator Precedence in Expression Evaluation?

Answer:

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated.

Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Arithmetic operator	* ,/ ,%, +, _	Left to right
Shift	<< ,>>	Left to right
Relational operator	< ,<= ,> ,>=	Left to right
Equality	==, !=	Left to right
Bitwise operator	&,^, |	Left to right
Logical operator	&&, ||	Left to right
Conditional	?:	Right to left

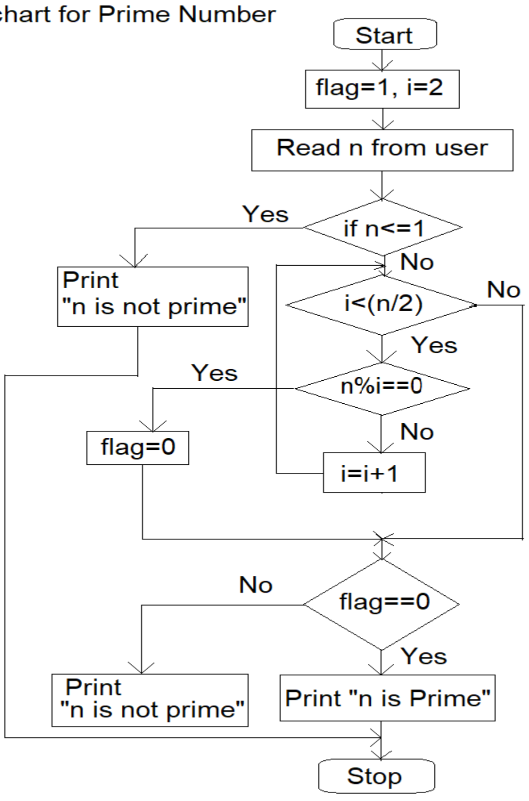
3. Develop Algorithm, Flowchart and program code for printing prime number?

Answer:

Algorithm:

```
step 1: Start
Step 2: Initialize variables num,flag=1, i=2
Step 3: Read num from user
Step 4: If num<=1
            Display "num is not a prime number"
            Goto step 7
Step 5: Repeat the steps until i<[(n/2)+1]
            5.1  If remainder of number divide i equals to 0,
                    Set flag=0
                    Goto step 6
            5.2  i=i+1
Step 6: If flag==0,
            Display “ prime number"
        Else
            Display “not  prime number"
Step 7: Stop
```

Flowchart for Prime Number



Program:

```
#include <stdio.h>
int main()
{
   int n, i, flag = 1;

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

   for(i=2; i<=n/2; ++i)
   {
      if(n%i==0)
      {
         flag=0;
         break;
      }
   }

   if (flag==0)
      printf("%d is a prime number.",n);
   else
      printf("%d is not a prime number.",n);

   return 0;
}
```

Output:

```
Enter a positive integer: 56
56 is not a prime number.
```

```
Enter a positive integer: 13
13 is a prime number.
```

4. Explain the iterative statements in C language with examples.

Answer:

- We have two types of looping statement
- One in which condition is tested before called entry control loop. **(for loop & while loop)**
- The other in which condition is checked at exit called exit controlled loop. **(do while loop)**

Step to be followed while working with loops:

• First initialization is done once,

• then condition is evaluated and if it is true then body of loop is executed.

• After execution of body the control goes to increment/ decrement part then condition is once again evaluated and if true body

While loop syntax:

```
while(test condition)
 {
 body of the loop
 }
```

for loop syntax:

```
for(initialization; test control; increment/decrement)
{
body of loop
 }
```

Do While loop syntax:

The while loop does not allow body to be executed if test condition is false. The do while is an exit controlled loop and its body is executed at least once.

```
do {
 body
 }while(test condition);
```

Note: write any suitable program

5. what is an opertor? Explain relational operator with example?

Answer:

The symbols which are used to perform logical and mathematical operations in a C program are called C operators.

These C operators join individual constants and variables to form expressions.

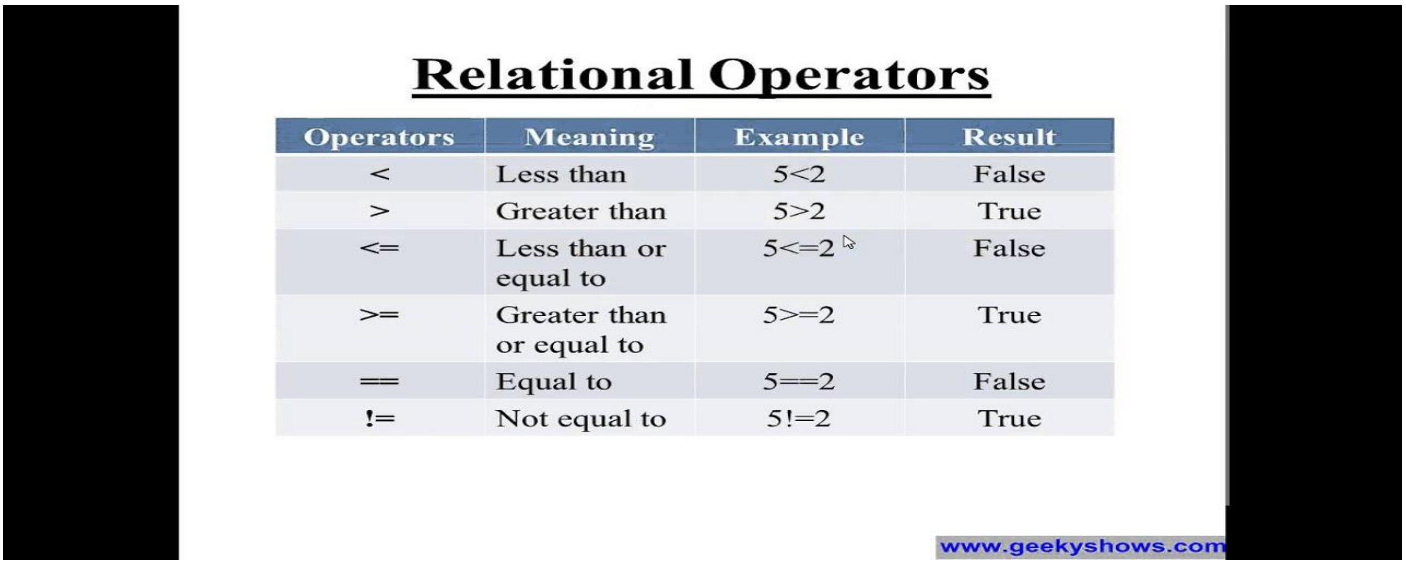
Operators, functions, constants and variables are combined together to form expressions

Relational operators:

• Relational operators in c programming is used for specifying the relation between two operands such as greater than, less than and equals.

• Relational operators are used to compare, logical, arithmetic and character expression and each relational operator takes two operands.

• Each operator compares their left side with their right side. It evaluates to 0 if the condition is false and 1 if it is true



Relational Operators

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

www.geekyshows.com

UNIT 2:

1.What is an array ? Different ways for initialising 1D array?

Answer:

An array is collection of items or elements stored at continuous memory locations.

Declaration:

datatype arrayname[maxsize];

Declaration and Initialization:

Storage class datatype arrayname[size] = {List of Value};

Different ways for initialising 1D array:

1. int arr[10]; // declaration for one dimensional array

2. int arr[] = {10, 20, 30, 40} // declaration and initialization above is same as

    int arr[4] = {10, 20, 30, 40}

3. int arr[6] = {10, 20, 30, 40} above is same as

    int arr[] = {10, 20, 30, 40, 0, 0}"

Memory Address of an array Elements are accessed by specifying the index (offset) of the desired element within square [ ] brackets after the array name.

2. List and explain any five string handling functions with examples.

Answer:

1 **strcpy(s1, s2)**;	Copies string s2 into string s1.
2 **strcat(s1, s2);**	Concatenates string s2 onto the end of string s1.
3 **strlen(s1);**	Returns the length of string s1.
4 **strcmp(s1, s2);**	Returns 0 if s1 and s2 are the same; less than 0 if s13.
5 **strchr(s1, ch**);	Returns a pointer to the first occurrence of character ch in string s1.
6 **strstr(s1, s2);**	Returns a pointer to the first occurrence of string s2 in string s1.

Program:

```
#include<stdio.h>
 #include <string.h>
int main ()
 {
        char str1[12] = "Hello";
         char str2[12] = "World";
        char str3[12];
        int len ;
        strcpy(str3, str1);
        printf("strcpy( str3, str1) : %s\n", str3 );
        strcat( str1, str2);
        printf("strcat( str1, str2): %s\n", str1 );
        len = strlen(str1);
         printf("strlen(str1) : %d\n", len );
        return 0;
         }
```

Output :

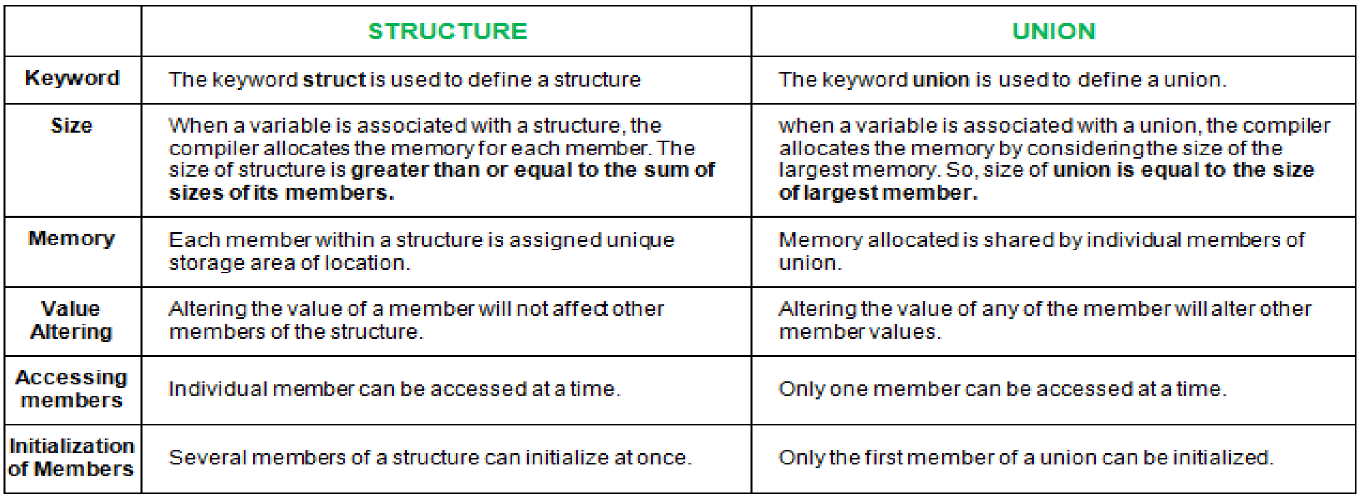
strcpy(str3, str1) : Hello

strcat(str1, str2) : HelloWorld

strlen(str1) : 10

3.Differentiate between structure and union?

<u>Answer:</u>



	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

Syntax	struct tagname
	{
	member 1;
	member 2;
	... member m;
	};
	struct tagname variable1, variable2....... ;
		union tagname
		{
		member 1;
		member 2;
		... member m;
		} ;
		struct tagname variable1, variable2....... ;

4. Define a structure for Student with Sno, Sname, marks of three subjects ,avg. Write a C program to read 4 students information calculate average and display 4 students information.

<u>Answer:</u>

PROGRAM:

```
#include<stdio.h>
struct student
{
            int sno;
            char sname[20];
            int mark[3];
             float avg;
};
```

```
int main()
{
            struct student  s[4];
             int i , j, sum=0;
            for(i=0;i<4;i++)
            {
                        sum=0;
                        printf(" enter  student %d details\n",i+1);
                        printf("sno,  sname\n");
                        scanf("%d%s", &s[i].sno, s[i].sname);
                        printf("enter 3 subject marks\n");
                        for(j=0;j<3;j++)
                        {
                              scanf ("%d",&s[i].mark[j];
                              sum=sum+ s[i].mark[j];
                        }
                        s[i].avg=sum/3.0;
            }
            Printf("Student details are");
            for(i=0;i<4;i++)
            {
                        printf("%d student : ",i+1);
                        printf("%d student sno: %d\n",s[i].sno);
                        printf("%d student name: %s\n",s[i].sname);
                        printf("%d student pecentage: %f\n",s[i].avg);

            }
return 0;
}
```

5. what is pointer ? explain pointer arithmetic.

<u>Answer:</u>

Pointers in C language is a variable that stores/points the address of another variable.

A Pointer in C is used to allocate memory dynamically i.e. at run time.

The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

Pointer is a derived data type.

Syntax:

Datatype *pointername;

Example :   int *p;   char *p;

pointer arithmetic: Operation perform on the pointers are

int * ptr, * p, a=20 , b=10;

p= &a;

ptr= &b;

1 poninter increment & decrement. (p++, ptr--, --p, ++ptr)

2 substraction of two pointer. (ptr – p)

3 poniter can be added or subtracted with constant integer value. (ptr + 3, p-4)

4 Relational operator : two pointer variable can be compared. (ptr< p, ptr==p)

5 Assignment operator: one pointer value can be assign to another pointer.(p= ptr)

UNIT 3:

1. Explain different preprocessor directives in C?

Answer:

Sr.No.	Directive & Description
1	**#define:** Substitutes a pre-processors macro. Example: #define max 7
2	**#include:** Inserts a particular header from another file.
3	**#undef :** Undefines a pre-processor macro.
4	**#ifdef:** Returns true if this macro is defined.
5	**#ifndef :** Returns true if this macro is not defined.
6	**#if:** Tests if a compile time condition is true.
7	**#else :** The alternative for #if.
8	**#elif :** #else and #if in one statement.
9	**#endif :** Ends preprocessor conditional.

Program:
```
#include <stdio.h>
#define PI 3.1415
#define circleArea(r) (PI*r*r)
int main()
{
    float radius, area,area1;
    printf("Enter the radius: ");
    scanf("%f", &radius);
    area = PI*radius*radius;
    printf("Area=%.2f",area);
    area1 = circleArea(radius);
    printf("Area = %.2f", area1);
     printf("Current time: %s",--TIME--);
    return 0;
```

}

OUTPUT:

```
Enter the radius: 5
Area=78.54
Area1 = 78.54
Current time: 07:19:33
```

2. define file? What are the types of file ?

Answer:

In C programming, file is a place on computer disk where information is stored permanently. Which represents a sequence of bytes ending with an end-of-file marker(EOF) .

Types of Files

1. Text files
2. Binary files

1. Text files

Text files are the normal .txt files that you can easily create using Notepad or any simple text editors.

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

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

2. Binary files

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

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

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

3. Define file pointer? Explain how to create,open, and read a file?

Answer:

Declaration for file Pointer:

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

Syntax:

FILE *file_pointer_name;

Example: FILE *fp;

Opening a File or Creating a File:

The fopen() function is used to create a new file or to open an existing file. this function available in stdio.h file

Syntax:

fp= fopen(“filename”, “mode”);

filename is the name of the file to be opened and **mode** specifies the purpose of opening the file. Mode can be of following types,

Example:

fp= fopen(“sample.txt”, “w”);

fp= fopen(“sample.txt”, “r”);

Reading and writing to a text file:

For reading and writing to a text file, we use the functions

Function	**Description**
getc()	reads a character from a file
putc()	writes a character to a file
fscanf()	reads a set of data from a file
fprintf()	writes a set of data to a file
getw()	reads a integer from a file
putw()	writes a integer to a file

4. Write a C program to merge two files into a third file (i.e., the contents of the first file followed by those of the second are put in the third file).

Answer:

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
   FILE *fs1, *fs2, *ft;
    char ch;

    fs1 = fopen(“file1.txt”,"r");
   fs2 = fopen(“file2.txt”,"r");
    ft = fopen(“file3.txt”,"w");
    if( fs1 == NULL || fs2 == NULL || ft == NULL )
```

```
  {
     printf("Press any key to exit...\n");
     exit(0);
  }

   while( ( ch = fgetc(fs1) ) != EOF )
            fputc(ch,ft);
   while( ( ch = fgetc(fs2) ) != EOF )
             fputc(ch,ft);
   printf("Two files were merged into third file successfully.\n");
   fclose(fs1);
  fclose(fs2);
  fclose(ft);
   return 0;
}
```

INPUT:

Source file : file1.txt : ABC

file2.txt : XYZPQR

file3.txt : ABC XYZPQR

OUTPUT: Two files were merged into third file successfully

5. what are different file positioning function available in c. (fseek, rwind ftell)

<u>Answer:</u>

There is no need to read each record sequentially, if we want to access a particular record. C supports these functions for random access file processing.

1. fseek()
2. ftell()
3. rewind()

fseek():
This function is used for seeking the pointer position in the file at the specified byte.
Syntax: fseek(file_pointer, displacement, pointer position);

OR

int fseek (FILE *fp, long num_bytes, int origin) ;

Where
file_pointer ---- It is the pointer which points to the file.

displacement ---- It is positive or negative.

This is the number of bytes which are skipped backward (if negative) or forward (if positive) from the current position.

Pointer position:

This sets the pointer position in the file.

Value	**pointer position**
0	Beginning of file.
1	Current position
2	End of file

Example:

1) **fseek(p,10L,0)**

0 means pointer position is on beginning of the file, from this statement pointer position is skipped 10 bytes from the beginning of the file.

2) **fseek(p,5L,1)**

1 means current position of the pointer position. From this statement pointer position is skipped 5 bytes forward from the current position.

3)fseek(p,-5L,1)

From this statement pointer position is skipped 5 bytes backward from the current position.

ftell():

This function returns the value of the current pointer position in the file. The value is count from the beginning of the file.

Syntax: ftell(fptr);

Where fptr is a file pointer.

rewind():

This function is used to move the file pointer to the beginning of the given file.
Syntax: rewind(fptr);

Where fptr is a file pointer.

Program for File position function

<u>**Answer:**</u>

```
#include
void main(){
  FILE *fp;
  int i;
  fp = fopen("file1.txt","r");
  for (i=1;i<5;i++){
     printf("%c : %d\n",getc(fp),ftell(fp));
     fseek(fp,ftell(fp),0);
```

```
    if (i == 5)
    rewind(fp);
  }
  fclose(fp);
}
```

UNIT 4:

1.What is function? Explain user defined function in details?

Answer:

A function is a **group of statements that together perform a task**.

Every C program has at least one function, which is main (),

user can divide up code into separate module called user defined functions.

the advantages of using user defined functions:

• Functions separate the concept (what is done?) from the implementation (how it is done?).

• Functions make programs easier to understand thus improves the readability of program.

• Functions can be called several times in the same program, allowing the code to be reused

Function Declaration / Prototype

Every function in C program should be declared before they are used.

Function declaration gives compiler information about function name, type of arguments to be passed and return type.

syntax: return_type function_name(type(1) argument(1),....,type(n) argument(n));

example: int add(int a, int b);

Function Call

Control of the program cannot be transferred to user-defined function unless it is called invoked.

Syntax : function_name(argument(1),....argument(n));

example: add(a, b);

Function Definition

Function definition contains programming codes to perform specific task.

Syntax:

```
return_type function_name(type(1) argument(1),..,type(n) argument(n))
 {
 //body of function
 }
```

Example: Program to Print a sentence using function.

```
#include <stdio.h>
void display(); //function declaration
void main()
 {
display(); //function call
}
```

```
void display() //function definition
  {
     printf("C Programming");
  }
```

Output: C Programming

2. Compare parameter passing techniques of Call by Value and Call by reference with an examples.

<u>Answer:</u>

Call by Value : In this type, value of actual arguments are passed to the formal arguments and the operation is done on the formal arguments. Any changes made in the formal arguments does not affect the actual arguments because formal arguments are photocopy of actual arguments. Changes made in the formal arguments are local to the block of calledfunction. Once control returns back to the calling-function the changes made vanish.

Example: Program to send values using call-by-value method.

```
#include <stdio.h>
Void swap(int  a, int b);
void main()
 {
 int x,y;
printf("enter values of x & y : ");
scanf("%d %d ", &x, &y);
printf("\n old values x=%d y =%d", x, y);
swap(x,y) ;
printf("\n new values x=%d y =%d", x, y);
 }
```

```
void swap(int a, int b)
{
        int t=a;
        a=b;
        b=t;
}
```

Output:

```
enter values of x & y : 2 3
old values x=2 y =3
new values x=2 y =3
```

Call by reference: The Address of actual parameters are copied to formal parameters. Here changing the formal parameters indirectly affects the actual parameters.

Example: Program to send values using call-by-reference method.

```
#include <stdio.h>
Void swap(int *a, int *b);
void main()
 {
 int x,y;
printf("enter values of x & y : ");
scanf("%d %d ", &x, &y);
printf("\n old values x=%d y =%d", x, y);
swap( &x, &y) ;
printf("\n new values x=%d y =%d", x, y);
 }
```

```
void swap(int  *a, int  *b)
{
        int t=*a;
        *a=*b;
        *b=t;
}
```

Output:

```
        enter values of x & y : 2 3
        old values x=2 y =3
        new values x=3 y =2
```

3. List and brief the uses of Dynamic Memory Allocation Functions. Write a C program to allocate a block of memory using malloc()?

Answer:

Dynamic Memory Allocation Functions:

Function	**Purpose**	**Syntax**
malloc()	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.	ptr =(data_type *) alloc(byte_size); Example: ptr = (int *) malloc (50)

calloc()	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.	ptr= (data_type *) calloc (n, size);
		Example:
		ptr =(int *)calloc(10, sizeof(int));

realloc()	It is used to modify the size of previously allocated memory space.	ptr = realloc (ptr,newsize);

Free()	Frees or empties the previously allocated memory space.	Free(ptr);

Advantages:

- Data structures can grow and shrink according to the requirement. (less memory wastage)
- We can allocate (create) additional storage whenever we need them.
- We can de-allocate (free/delete) dynamic space whenever we are done with them.
- Dynamic Allocation is done at run time.

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

Output

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

4. write recursive & non recursive program to calculate factorial of a number.

```
void main()
{
            int n;
           printf("enter the  number :");
            scanf("%d",&n);
            printf("factoria of number using function %d",fact(n));
            printf("factoria of number using recursion %d",rec_fact(n));
}
```

```
int fact(int n)   //sub program
 {
            int f=1,i;
            if((n==0)||(n==1))
                    return(1);
```

```
int rec_fact(int n)   //sub program
          {
                   if((n==0)||(n==1))
                   return  1;
                   else
```

```
        else                                                    return  n* rec_fact(n-1);
                for(i=1;i<=n;i++)                                  }
                f=f*i;
        return(f);
 }
```

Output:

```
        enter the  number :  5
        factoria of number using function:  120
        factoria of number using recursion:  120
```

UNIT 5:

1.Build Linear Search on list of elements 20, 12, 17, 28, 70, 80 and key element is 70.

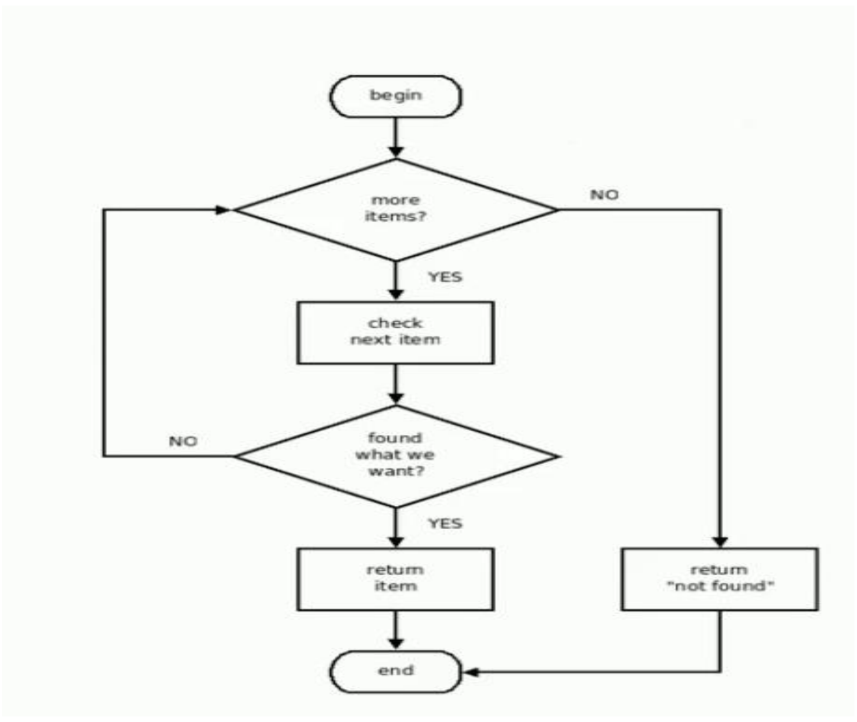
Write the algorithm, draw the flowchart along with the program.

Answer:

Algorithm:

```
1. variables needed: array[100], key, i, n, found=0
2.  read n value
3.   enter n elements into array
4.   read the value for search key.
5.   for (i = 0 ;i < n ; i++ )
        {
         if (a[i] == key)
                {
                 found=1; break;
                }
         }
6. if found = = 1,
         6.1 Display message "Search is successful and item is found at location:  loc"
7. else
        7.1 Display the message "Search Unsuccessful" and exit.
8. stop
```

Flowchart:



Program:

```
#include <stdio.h>

int main()
{
  int array[100], key, i, n, found=0;

  printf("Input number of elements in array\n");
  scanf("%d", &n);

  printf("Input %d numbers\n", n);

  for (i = 0; i < n; i++)
    scanf("%d", &array[i]);

  printf("Input a number to search\n");
  scanf("%d", &key);

  for (i = 0 ;i < n ; i++ )
 {
    if (a[i] == key)
          {
           found=1; break;
          }
 }
  if (found==0)
    printf("%d isn't present in the array.\n", key);
  else
    printf("%d is present at location %d.\n", key, i+1);

  return 0;
}
```

Output:

```
Enter the number of elements in array

5

Enter 5 numbers

2

6

18

25

1

Enter the number to search

25

25 is present at location 4.
```

2. write a program for bubble sort.

Answer:

Program:

```
#include<stdio.h>
void main()
{
 int a[50],n,i,j,temp;
 printf("Enter the size of array: ");
 scanf("%d",&n);
 printf("Enter the array elements: ");
  for(i=0;i<n;++i)
  scanf("%d",&a[i]);
  for(i=1;i<n;++i)
            for(j=0;j<(n-i);++j)
            if(a[j]>a[j+1])
            {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
            }
  printf("Array after sorting: ");
 for(i=0;i<n;++i)
```

```
  printf("%d ",a[i]);

}
```

OUTPUT:

```
Enter total elements: 5
Enter 5 elements: 45 67 32 68 20

 The array after sorting is:  20 32 45 67 68.
```