1. **What is a string? Write short notes on different string handling functions**

Ans:-stings in c language are a type of array specifically they are a character array . strings are collection of char values. All strings end in a '0' that 0 tells c where the string has ended.

String handling functions in c are:-

To perform manipulations on string data , there are built in function (library function) supported by 'c' compiler. the string functions are

- STRLEN(S1)
- STRCMP(S1,S2)
- STRCPY(S1,S2)
- STRCAT(S1,S2)

STRLEN :- this function is used to return the length of the string.

Name s1,

Ex;S1='MOID'

STRLEN(S1)=4

STRCMP (S1,S2):- this is a library function used to compare two strings .this function returns value <zero when s1 is less than s2 and 0 when s1 = s2 and will return >0 when s2 >s1.

STRCPY:- this function is used to copy string s2 in the string s1

STRCAT:- this library function is used to join two strings one after the other .this function concatenates s2 at the end of s1.

2. **Differentiate between recursion and iteration**

Ans:- Recursion: It is a process by which a function calls itself repeatedly until some base condition is reached. A function that calls itself is known as a recursive function. And, this technique is known as recursion. When a recursive function is executed the recursive function calls are placed on a data structure called stack until some base condition is reached as the recursive calls are not executed immediately.

Points to remember:

1. When using recursion, loops must be omitted.

2. As we don't have immediate answers in the recursive programs, the recursive calls are placed on stacks.

3. Stack is a data structure used to implement recursive programs.

Advantages:

1. Sometimes easy to solve using recursion rather than iterative solution.

2. Easy solution for a recursively defined problem.

Disadvantages:

1. Solution is always logically difficult to trace.
2. Takes a lot of space.
3. Uses more processor time.

The iterations in the C language are the statements which are executed number of times until a certain condition is reached. These iterations are regarded as "**Loops**" in C language. The Iterations can be able to modify the control flow of the program thus they referred as "Control Statements."

3. Differentiate actual parameters with formal parameters.

Arguments: argument refers to the variable passed to the function. They are of two types:

(a) **Actual:** The arguments that are passed in a function call are called actual arguments. These arguments are defined in the calling function.

(b**) Formal**: Formal Parameters are the parameters which are used in the called function.

Distinguishing between local and global variables:

Local variables	Global variables
These are declared within the body of the function.	These are declared outside the function.
These variables can be referred only within the function in which it is declared.	These variables can be referred from any part of the program.
The value of the variables disappear once the function finishes its execution.	The value of the variables disappear only after the entire execution of the program.

3. Define structure. How do you access structure members / elements?

Structures

Arrays allow to define types of variables that can hold several data items of the same kind. Similarly structure is a user defined data type where variables of different data types are grouped under a single name.

Structure members are accessed using dot (.) operator also called period operator or member access operator. **Syntax: struct varname.memberName;**

4. Write a C program to implement binary search.

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

```
int i, low, high, mid, n, key, array[100];
printf ("Enter number of elements\n");
scanf ("%d", &n);
printf ("Enter %d integers\n", n);
for (i = 0; i < n; i++)
scanf ("%d", &array[i]);
printf ("Enter value to find\n");
scanf ("%d", &key);
low = 0;
high = n - 1;
mid = (low + high) / 2;
while (low <= high)
{
if (array[mid] < key)
low = mid + 1;
else if (array[mid] == key)
{
printf ("%d found at location %d\n", key, mid + 1);
break;
}
else
high = mid - 1;
mid = (low + high) / 2;
}
if (low > high)
printf ("Not found! %d isn't present in the list\n", key);
return 0;
}
```

5. **What is a pointer variable? Write its advantages.**
   Ans:-pointer variable is a variable which stores the address of the other variable.
   I.e pointer variables contain address (location) of the other variables as there values.
   If one variable contains the address of the other variable then the first one is said to be the pointer of the other .
   Syntax:- data type *ptr var name;

7. Explain how arrays are passed to a function with an example.

We can pass a single element of an array as an argument to function using its index value.

Example:

```
void display(int);
main( )
{
Int a[4]={10,20,30,40};
display(a[3]);
}
void display(int x)
{
printf("\n The element accessed is %d", x);
}
```

The element at index a[3] i.e., 40 is passed to display(), so it displays 40

8. Create a structure student with roll no, name and marks of 3 subjects. Write a C program to display roll no, name and average marks using an array of structures.

```
#include<stdio.h>
main()
{
int m1, m2, m3,avg;
printf("Enter the marks of 3 subjects\n");
scanf("%d %d %d",&m1, &m2, &m3);
avg=(m1+m2+m3)/3;
if(avg>=90 && avg<=100)
printf("Grade A");
else if(avg>=80 && avg<=89)
printf("Grade B");
else if(avg>=70 && avg<=79)
printf("Grade C");
else if(avg>=60 && avg<=69)
printf("Grade D");
```

```
else if(avg>=50 && avg<=59)

printf("Grade E");

else if(avg>=40 && avg<=49)

printf("Grade F");

else

printf("FAIL");

}
```

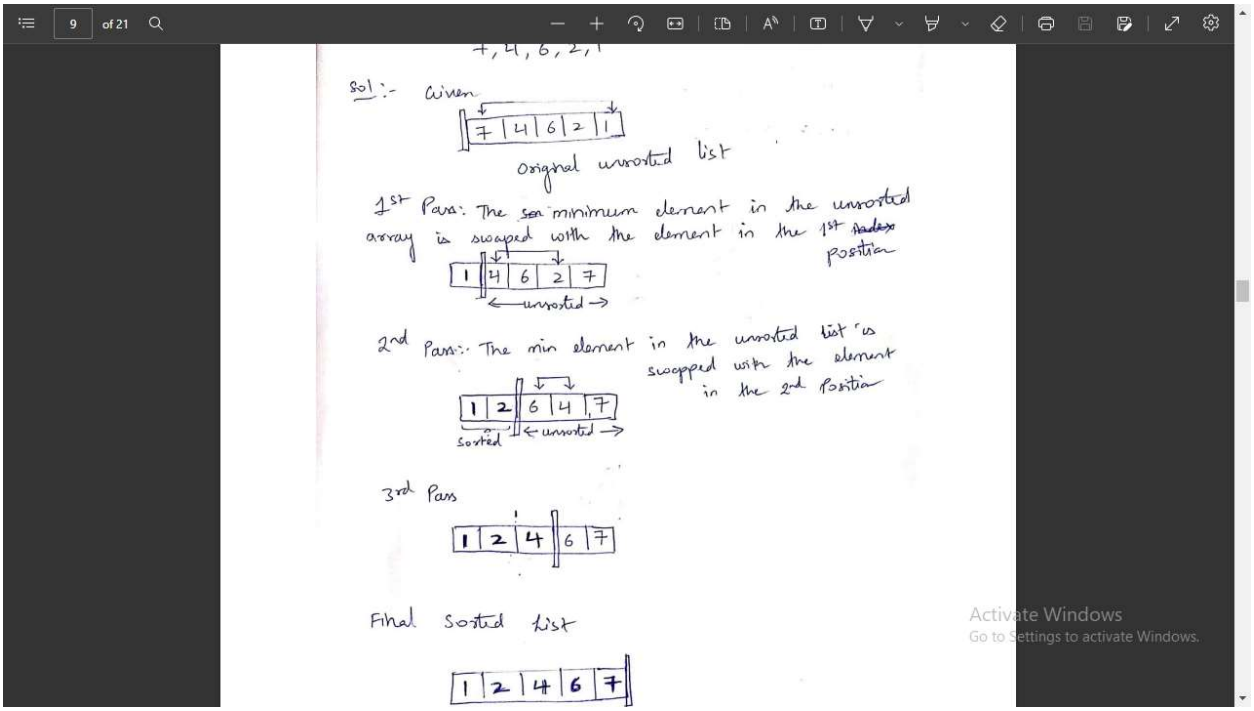
9. How to use pointers in arguments in a function? Explain with a program(skipping)

10.What is the purpose of main() in C? Is it a predefined or user-defined function? Justify it with a proper example.

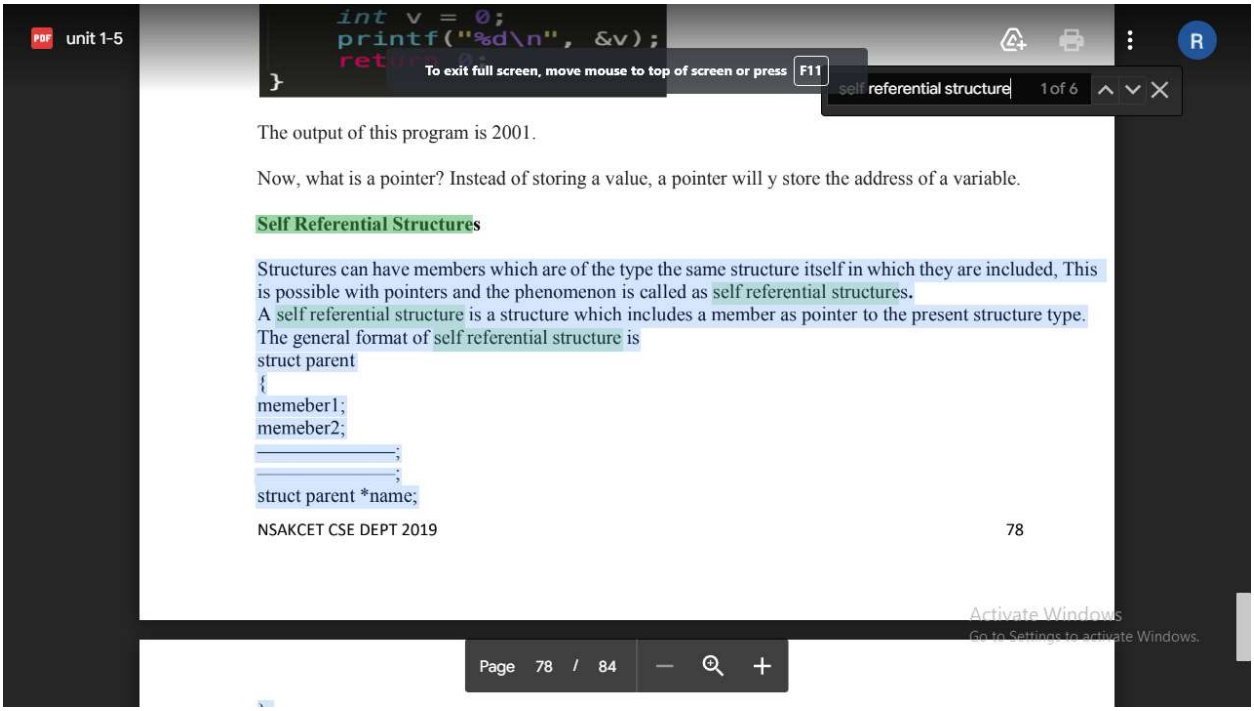
A **main** () function is a user-defined function in **C** that means we can pass parameters to the main () function according to the requirement of a program. A **main** () **function** is used to invoke the programming code at the run time, not at the compile time of a program. A **main** () function is followed by opening and closing parenthesis brackets.

11. Explain selection sort with an example.

Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.



12.Explain self referential structure with a C program.



Example

```
struct element
{
char name{20};
int num;
struct element * value;
}
```

13.What is a file? Explain different file handling functions.

File Handling:

1. New way of dealing with **data is file handling**.
2. Data is **stored onto the disk** and can be retrieve whenever require.
3. Output of the program may be stored onto the disk
4. In C we have many **functions that deals with file handling**
5. A file is a collection of bytes stored on a **secondary storage device** (generally a disk)
6. Collection of byte may be **interpreted as** –
   - Single character
   - Single Word
   - Single Line
   - Complete Structure.

Modes of open

The file can be open in three different ways as

Read mode ' r'/rt

Write mode 'w'/wt

Appened Mode 'a'/at

Reading a character from a file

getc() is used to read a character into a file

Syntax:

character_variable=getc(file_ptr);

Writing acharacter into a file

putc() is used to write a character into a file

puts(character-var,file-ptr);

ClOSING A FILE

fclose() function close a file.

fclose(file-ptr);

fcloseall () is used to close all the opened file at a time

File Operation

14.Recursive programs to implement

● Factorial

● Fibonacci series up to given limit

```
#include<stdio.h>
void fibonacciSeries(int range)
{
int a=0, b=1, c;
while (a<=range)
{
printf("%d\t", a);
c = a+b;
a = b;
b = c;
}
```

```
}
int main()
{
int range;
printf("Enter range: ");
scanf("%d", &range);
printf("The fibonacci series is: \n");
fibonacciSeries(range);
return 0;
}

#include <stdio.h>
int factorial(int i)
{
if(i <= 1)
{
return 1;
}
return i * factorial(i - 1);
}
int main() {
int n;
printf("Enter number\n");
scanf("%d",&n);
printf("Factorial of %d is %d\n", n, factorial(n));
return 0;
}
```

15.What are the different types of parameter passing techniques?

1. Call by value:

● In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

```
#include void update(int p) {

 printf("Before updating value inside function p=%d \n",p);

p=p-2;

printf("After updating value inside function p=%d \n", p);

}

int main()

{ int x=10;

printf("Before function call x=%d \n", x);

update(x);//passing value in function

printf("After function call x=%d \n", x);

return 0;

}
```

2. Call by reference

● In call by reference, the address of the variable is passed into the function call as the actual parameter. The address is stored using a special variable called pointer.

● Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

```
#include void update(int *p)

{

printf("Before updating value inside function num=%d \n",*p);

*p=*p-2;

printf("After updating value inside function num=%d \n", *p);

}

int main()

{ int x=10;

printf("Before function call x=%d \n", x);
```

```
 update(&x);

//passing value in function

printf("After function call x=%d \n", x);

return 0;

}
```

16.How are unions different from structures?

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

(You have already studied what is structure.) write on your own

17. 2D array programs

● Matrix Addition

● Matrix Multiplication

```
#include<stdio.h>

int main()

{

int a[20][20],b[20][20],c[20][20];

int i,j,n,m;

printf("Enter the rows and column\n");

scanf("%d%d",&n,&m);

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

{

for(j=0;j<m;j++)

{

printf("Enter the elements a[%d][%d]",i,j);

scanf("%d",&a[i][j]);

}

}
```

```
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
printf("Enter the elements b[%d][%d]",i,j);
scanf("%d",&b[i][j]);
}
}
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}
printf("\nAddition of matrix is \n");
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}}

//Program for matrix multiplication
#include<stdio.h>
#include<stdlib.h>
int main(){
```

```
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
system("cls");
printf("enter the number of row=");
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
printf("enter the first matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
```

```
{
mul[i][j]+=a[i][k]*b[k][j];
}
}
}
//for printing result
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
return 0;
}
```

18.What is a function prototype? How is it different from function definition?

A function prototype is simply the declaration of a function that specifies the function's name, parameters and return type. It is the first line after the preprocessor directives and must terminate with a semicolon.

Syntax of function prototype

returnType functionName(datatype of arg1, datatype of arg2, ...);

example, int addNumbers(int a, int b); is the function prototype which provides the following information to the compiler:

1. name of the function is addNumbers()

2. return type of the function is int

3. two arguments of type int are passed to the function

The function prototype is not needed if the user-defined function is defined before the main() function