Computer programming using c

Important questions

Short Answer Questions

1. Define number system? What are the different types of number systems?

A.

Number System: It is a way to represent the numbers in particular formats

There are 4 number systems:

Binary number system.

Octal number system.

Decimal number system.

Hexa decimal number system.

2. Define Algorithm and what are the properties of an algorithm?

A.

An algorithm is a finite set of steps defining the solution of a particular problem. .

The characteristics of an algorithm are as follows:

a. Input
b. Output
c. Finiteness
d. Definiteness
e. Effectiveness

3. What is conditional operator or ternary operator?

A.

The Conditional Operator in C, also called a Ternary operator, is one of the Operators, which used in the decision-making process. The C Programming Conditional Operator returns the statement depends upon the given expression result.

syntax: Test_expression ? statement1: statement2

4. Define array? Write the syntax for declaring a one dimensional array?

A.

Array is a collection of homogeneous elements.

syntax: datatype arrayname[size];

Here size represents number of elements that array can store.

5. Define a function? Write the syntax for declaring a user defined function?

A.

Function is a set of statements used to perform a specific task.

syntax:

```
return type function name(argument list)
{
statements;
}
```

6. Define pointer? Write the syntax for declaring a pointer?

A.

Pointer is a variable that stores the address of another variable of same data type.

Syntax for declaring a pointer:

data type *pointer name;

Ex: int *p; // p is an integer pointer, that can store the address of an integer variable.

7. What is pointer to pointer?

A.

A pointer which the store the address of an another pointer is known as pointer to pointer.

syntax:

datatype **pointername;

8. Define a string?

A.

Collection of characters ends with null character(\0).

syntax:

char string name[size];

9. List any two string manipulation functions with an example?

A.

String Manipulation functions:

The action of the fundamental operations on strings, which includes creation, concatenation and extraction of string segments.

They are included in string.h header file.

1 strlen()

2 strcpy()

3 strcat()

4 strcmp() etc.

10. Difference between Structure and union?

A.
There are two major differences between structure and union.
First, in structure memory is allocated for all members, while in union memory will be allocated for the largest member only.
Second, a union may only be initialized with a value of the type of its first member.

11. Define: a. enum b. typedef c.bit-fields.

A.

enum:

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

```
enum State {Working = 1, Failed = 0};
```

typedef:

The C programming language provides a keyword called **typedef**, which you can use to give a type a new name or allias name.

syntax: typedef oldname newname

bit-fields:

Bit fields are used in structures. In C, we can specify size (in bits) of structure and union members. The idea is to use memory efficiently when we know that the value of a field or group of fields will never exceed a limit or is within a small range.

```
#include <stdio.h>

struct date

{
```

```
unsigned int d:5;

unsigned int m:4;

unsigned int y;

};

void main()

{

printf("Size of date is %d bytes\n", sizeof(struct date));

struct date dt = {31, 12, 2014};

printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);

}
```

12. List stack/queue operations.

A.

Stack is a linear data structure, which can store data in a linear fashion LIFO method.

Operations:

push(), pop(), display(),underflow(),overflow()

Queue is a linear data structure, which can store data in a linear fashion FIFO method.

Operations:

enqueue(), dequeue(), display()

13. Define File? Types of files.

A.

A file represents a sequence of bytes on the disk where a group of related data is stored.

We have two types of files:--

1.Text file

2. Binary file

14. What is the use of fseek ()?

A.

fseek( ) is a file positioning function, which takes file pointer to the desired location of your file.

Syntax:-

```
fseek(file_pointer, offset_value, position_value);
```

15. What is the use of ftell ()?

A.

ftell()= returns the current position of file pointer.

syntax:

Syntax : int n = ftell (file pointer)

16. List any 2 differences between text file and binary file.

A.

1. Text file is human readable because everything is stored in terms of text. In binary file everything is written in terms of 0 and 1, therefore binary file is not human readable.

2. A newline(\n) character is converted into the carriage return-linefeed combination before being written to the disk. In binary file, these conversions will not take place.
3. In text file, a special character, whose ASCII value is 26, is inserted after the last character in the file to mark the end of file. There is no such special character present in the binary mode files to mark the end of file.
4. In text file, the text and characters are stored one character per byte. For example, the integer value 1245 will occupy 2 bytes in memory but it will occupy 5 bytes in text file. In binary file, the integer value 1245 will occupy 2 bytes in memory as well as in file.

17. Differentiate while and do-while statements.

A.

while loop	**do while loop**
It is a entry control loop	It is a exit control loop
Statements will be executed 0 or more times	Statements will be executed 1 or more times
Condition will be checked first then statements will be executed.	Statements will be executed first then the condition will be checked.
If there is a single statement, brackets are not required.	brackets are always required.

18. What are the selection statements?

A.

The statements which are used to place the conditions in a c program to control the flow of execution of a c program are known as selection statements.

In c language the following are the selection statements

- simple if
- nested if
- if else
- else if ladder

19. What are the iterative statements used in C?

A.

The statements which are used to repeat a statement or set of statements inside a c program until the condition fails are known as iterative or looping statements.

In c language the following are the iterative statements

- for
- while
- do while

20. What are the different Unconditional control statements?

A.

These are unconditional control statements.

- ❖ goto transfers the control to desired location in the program.
- ❖ break is used to break the flow of control.
- ❖ continue is used to repeat the group of statements by skipping the statements after continue keyword.

Ex:-

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

```
void main()
{
        int n;
        stmt:
        printf("%d",n);
        n++;
        if(n<=5)
                gotostmt;
        for(n=1;n<=10;n++)
        {
                printf("%d",n);
                if(n==5)
                break;
        }
        n=1;
        while(n<=10)
        {
                if(n==5)
                continue;
                printf("%d",n);
        }
        printf("End of the program");
}
```

21. Define variable?

A.

variable is a named memory location to store data of its type.

syntax:

datatype variablename;

22. Define an expression?

A.

Expression is collection of operators and operands.

a+b+c=0;

here a,b,c,0 are operands and +,= are operators.

1. In programming, an expression is any legal combination of symbols that represents a value.
2. C Programming Provides its own rules of Expression, whether it is legal expression or illegal expression.
3. Every expression consists of at least one operand and can have one or more operators.
4. Operands are values and Operators are symbols that represent particular actions.

Expressions are classified as

1. Infix Expression.
2. Postfix Expression.
3. Prefix Expression.

23. Write the syntax for declaring a 2-D array?

A.

Syntax for declaring two dimensional array:

data type array_name [rows][cols]; Ex: int a[3][3];

24. What are self-referential structures?

A.

self-referential structures:

Self-Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.

```
struct node {
        int data1;
        char data2;
        struct node* link;
};
int main()
{
        struct node ob;
        return 0;
}
```

Long Answer Questions

1. Define algorithm? Write an algorithm for finding the greatest of 3 numbers.

A.

An algorithm is a finite set of steps defining the solution of a particular problem. .

The characteristics of an algorithm are as follows:

a. Input
b. Output
c. Finiteness
d. Definiteness
e. Effectiveness

Algorithm for finding the greatest of 3 numbers.

```
step 1:start
step 2:input a, b, c
step 3:if a>bgo to step 4,otherwise go to step 5
step 4:if a>c go to step 6,otherwise go to step 8
step 5:ifb>c go to step 7,otherwise go to step 8
step 6:output "a is the largest ",go to step 9
Start 7 : Output "b is the largest", goto step 9
Start 8 : Output " c is the largest", goto step 9
Start 9 : Stop
```

2. Briefly discuss about computer languages?

A.

Computer language is a language that has been used to communicate with the computer. Basically, these languages are classified into 3 kinds.

1. Low-level or Machine level Languages.

2. Middle-level or Assembly level languages.

3. High-level languages.

1. Low-level or Machine level Languages:

   A **low-level language** is a programming language that provides little or no abstraction of programming concepts and is very close to writing actual machine instructions. Two examples of low-level languages are assembly and machine code.

2. Middle-level or Assembly level languages:

   The middle-level language lies in between the low level and high-level language. C language is the middle-level language. By using the C language, the user is capable of doing the system programming for writing operating system as well as application programming. The Java and C++ are also middle-level languages.

3. High-level languages:

   A high-level language is a programming language designed to simplify computer programming. It is "high-level" since it is several steps removed from the actual code run on a computer's processor. High-level source code contains easy-to-read syntax that is later converted into a low-level language, which can be recognized and run by a specific CPU.

3. What are the different types of operators used in C language?

A.

Operator:

Operator in C is a symbol that tells the computer to perform mathematical or logical manipulation on data.

Different operators are:

1. Arithmetic Operators (+, -, *, /, %)
2. Increment and Decrement Operators (++, --)
3. Assignment Operators (=, +=, -=, *=, /=, %=)

4. Relational Operators (==,>,<,!=,>=,<=)
5. Logical Operators (&&,||,!)
6. Conditional Operators (?:)
7. Bitwise Operators(&,|,~,^,<<,>>)
8. Special Operators (comma operator, sizeof operator)

Examples:

//arithmetic operators

c=a+b;

c=a-b;

c=a*b;

c=a/b;

//Increment and decrement operators

c++;

c--;

++c;

--c;

//Assignment operators

a=b;

a+=b;

a-=b;

a*=b;

//Relational Operators

4==3 returns 0

4>3 returns 1

4<3 returns 0

4!=3 returns 1

4>=3 returns 1

4<=3 returns 0

//Logical operators

if c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.

if c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.

if c = 5 then, expression ! (c == 5) equals to 0.

4. Briefly discuss about conditional or selection statements used in C language?

A.

conditional statements or selection statements requires that the programmer specifies one or more conditions to be evaluated or tested by the program along with statements to be executed, if the condition is determined to be true & Optionally, if the condition is false other statements will be executed.

if statement

if statement is the simplest form of selection constructs. It is used to execute or skip a statement or statements by checking a condition. The condition is given as a relational expression. If the condition is true, the statement or set of statements after **if statement** is executed otherwise not executed.

Syntax:
The syntax of if statement is:

```
if (condition)
statement;
```

The above is for single statement. A set of statements can also be made conditional by writing the statements in braces { }. A set of statement is also called block of statements. The syntax for block of statements in if statements is:

```
if (condition)
{
statement 1;
statement 2;
.
.
statement n;

}
```

if else statement

It executes one statement or block of statements when the condition is **true** but if condition is **false** then it will execute another statement or block of statements. In any situation, one block is executed & the other is skipped.

Syntax:

The syntax for single statement:

```
if (condition)
statement;

else

statement;
```

The syntax for Block of statements:

```
if (condition)
{
statement 1;
statement 2;
.
.
statement n;

}
else
{
```

```
statement 1;
statement 2;
.
.
statement n;
}
```

Switch Statement

Switch statement is used for multiple choice or selection. It is used as a substitute of if-else statements. It is used when multiple choice are given & one choice is to be selected.

Syntax

```
switch(expression)

{

case const-1:

statements;

break;

case const-2:

statement;

break;

.

.

case const-n:

statement;

break;
```

default:

statements:

}

5. Briefly discuss about looping or iterative statements?

A.

The looping statements in C language are also known as Repetition statements or Iterative statements. They are as follows.

1. while loop
2. do-while loop
3. for loop

Syntax:

1. while loop

```
Initialization…

while(condition)

{

        //body of the loop

loop update→ increment or decrement

}
```

2. do while loop

```
Initialization..
do
{
    //body of the loop
    loop update
}while(condition);
```

3. for loop

```
for(Initialization ;condition ; loop update)
{
        //body of the loop
}
```

6. Define data type? What are the different data types supported in C language?

A.

Data types specify how we enter data into our programs and what type of data we enter. C language has some predefined set of data types to handle various kinds of data that we can use in our program. These datatypes have different storage capacities.

C language supports 2 different type of data types:

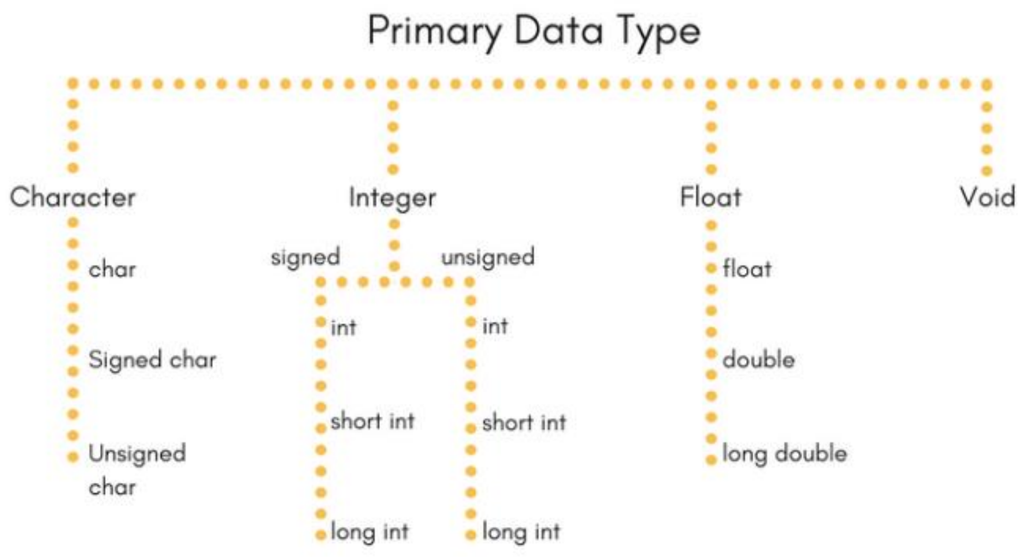
1. **Primary data types**:

   These are fundamental data types in C namely integer(int), floating point(float), character(char) and void.

2. **Derived data types**:

   Derived data types are nothing but primary datatypes but a little twisted or grouped together like **array**, **stucture**, **union** and **pointer**. These are discussed in details later.

Data type determines the type of data a variable will hold. If a variable x is declared as int. it means x can hold only integer values. Every variable which is used in the program must be declared as what data-type it is.



Integer type

Integers are used to store whole numbers.

Size and range of Integer type on 16-bit machine:

Type	**Size(bytes)**	**Range**
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

Floating point type

Floating types are used to store real numbers.

Size and range of Integer type on 16-bit machine

Type	**Size(bytes)**	**Range**

Float	4	3.4E-38 to 3.4E+38
Double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Character type

Character types are used to store characters value.**Size and range of Integer type on 16-bit machine**

Type	**Size(bytes)**	**Range**
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

void type

void type means no value. This is usually used to specify the type of functions which returns nothing. We will get acquainted to this datatype as we start learning more advanced topics in C language, like functions, pointers etc.

7. Define searching? Illustrate linear search with an example.

A.

Searching: It is a process of finding a key value in a given list of elements.

Input data:

90	85	83	79	74	72	67	52	48

Key: 74

Linear Search:

Here key element is compared with the first element of the list, if it matches that is the element you are looking for, if it does not match with it, compare with the next element of the list.

90!=74

85!=74

83!=74

79!=74

74==74

8. Define searching? Illustrate binary search with an example.

A.

Searching: It is a process of finding a key value in a given list of elements.

Binary Search:

90 85 83 79 74 72 67 52 48← List

Sort the given list of integers in increasing order.

48	52	67	72	74	79	83	85	90
0	1	2	3	4	5	6	7	8← Element Position

set the first element of the list to Low and last element to High

Now calculate mid for the list

In the above list, mid= low+high/2, mid=(0+8)/2=4

now compare key with mid, if(key==mid) => Element found at mid position.

if the key is less than mid, then high=mid-1.

if the key is greater than mid, then low=mid+1.

In the above example key found at mid, so element found at 4th position of the list.

9. Define sorting? Illustrate bubble sort with an example.

A.

Sorting:

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order.

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Example:

First Pass:

(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.

(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4

(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2

(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

(**1 4** 2 5 8) –> (**1 4** 2 5 8)

(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) –> (1 2 **4 5** 8)

(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Third Pass:

(**1 2** 4 5 8) –> (**1 2** 4 5 8)

(1 **2 4** 5 8) –> (1 **2 4** 5 8)

(1 2 **4 5** 8) –> (1 2 **4 5** 8)

(1 2 4 **5 8**) –> (1 2 4 **5 8**)

10. Write a C program to perform the following.

i) Addition of two matrices.

ii) Multiplication of two matrices.

A.

(i) addition of two matrices

```
#include<stdio.h>
#include<conio.h>
void add(int[][10],int[][10],int,int);
void main()
{
int a[10][10],b[10][10],i,rows,j,cols;
clrscr();
   printf("\nEnter No. of rows and columns:");
   scanf("%d%d",&rows,&cols);
   printf("\nEnter %d elements for the first %d x %d        matrix:",rows*cols,rows,cols);
   for(i=0;i<rows;i++)
   {
      for(j=0;j<cols;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
   printf("\nEnter %d elements for the second %d x %d matrix:",rows*cols,rows,cols);
```

```
    for(i=0;i<rows;i++)
    {
        for(j=0;j<cols;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    printf("\n first matrix is:\n");
    for(i=0;i<rows;i++)
    {
        for(j=0;j<cols;j++)
        {
            printf("%2d",a[i][j]);
        }
        printf("\n");
    }
    printf("\n second matrix is:\n");
for(i=0;i<rows;i++)
    {
        for(j=0;j<cols;j++)
        {
                printf("%2d",b[i][j]);
        }
        printf("\n");
    }
    add(a,b,rows,cols);//Function calling
}

void add(int a[][10],int b[][10],introws,int cols)// Function Definition
{
inti,j;
```

```
    printf("\n sum of first and second matrices is:\n");
    for(i=0;i<rows;i++)
    {
        for(j=0;j<cols;j++)
        {
            printf("%2d",a[i][j]+b[i][j]);
        }
        printf("\n");
    }

}
```

Output:

```
        Enter No. of rows and columns: 2 2


        Enter 4 elements for the first 2 x 2 matrix: 1 2 3 4


        Enter 4 elements for the second 2 x 2 matrix: 1 2 3 4


        first matrix is:
        1       2
        3       4


        second matrix is:
        1       2
        3       4


        sum of first and second matrices is:
        2       4
        6       8
```

(ii) multiplication of two matrices

```
#include<stdio.h>
#include<conio.h>
//FUNCTION DECLARATION
intmul(int[][10],int [][10],int,int,int,int);
void main()
{
int a[10][10],b[10][10],i,r1,r2,j,c1,c2;
clrscr();
   printf("\nEnter No.of rows and columns for the first matrix:");
   scanf("%d%d",&r1,&c1);//m1=rows,n1=columns of matrix 1
   printf("\nEnter %d elements for the first %dx%d matrix:",r1*c1,r1,c1);
   for(i=0;i<r1;i++)//READING ELEMENTS FOR MATRIX A
   {
      for(j=0;j<c1;j++)
      {
          scanf("%d",&a[i][j]);
      }
   }
   printf("\nEnter no. of rows and columns for the second matrix:" );
   scanf("%d%d",&r2,&c2);//m2=rows,n2=columns of matrix 2
if(r1==c2)//CHECKING THE MATRIX MULTIPLICATION RULE.
{
   printf("\nEnter %d elements for the second %dx%d martix:",r2*c2,r2,c2);
   for(i=0;i<r2;i++)
   {
      for(j=0;j<c2;j++)
      {
         scanf("%d",&b[i][j]);
```

```
        }
    }
    printf("\n first matrix is:\n");
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c1;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }
    printf("\n second matrix is:\n");
    for(i=0;i<r2;i++)
    {
        for(j=0;j<c2;j++)
        {
            printf("%d\t",b[i][j]);
        }
        printf("\n");
    }
mul(a,b,r1,r2,c1,c2);//CALLING MULTIPLICATION MATRIX
}
else
{
printf("\n no. of rows in the 1st  matrix must be same as the no. of cols in the 2nd matrix:");
}
getch();
}


intmul(int a[][10],int b[][10],int r1,int r2,int c1,int c2)//FUNCTION DEFINITION
```

```
{
inti,j,k,c[10][10];
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c2;j++)
        {
            c[i][j]=0;
            for(k=0;k<c1;k++)
            {
                c[i][j]=c[i][j]+a[i][k]*b[k][j];
            }
    }
    printf("\n product of two matrices is:\n");
    for(i=0;i<r1;i++)
    {
    for(j=0;j<c2;j++)
        {
            printf("%2d\t",c[i][j]);//PRINTING RESULTANT MATRIX
        }
        printf("\n");
    }
}
}
```

Output:

```
Enter No.of rows and columns for the first matrix:2 3


Enter 6 elements for the first 2x3 matrix:1 2 3 4 5 6


Enter no. of rows and columns for the second matrix:3 2


Enter 6 elements for the second 3x2 martix:1 2 3 4 5 6
```

```
 first matrix is:
1	2	3
4	5	6

second matrix is:
1	2
3	4
5	6

 product of two matrices is:
22	28
49	64
```

11. Define Function? Explain four different categories of user defined functions in C language.

A.

The different categories of functions depending on arguments and return type are classified as follows:

1. Function without return type and without arguments.
2. Function with return type and without arguments.
3. Function without return type and with arguments.
4. Function with return type and with arguments.

Examples for each of the categories

1. Function without return type and without arguments.

```
void print()
{
    printf(“ Hello”);
}
```

2. Function with return type and without arguments
   ```
   int get(void)
   {
       return(10);
   }
   ```
3. Function without return type and with arguments
   ```
   void getNum(int n)
   {
       printf(“number:%d”,n);
   }
   ```
4. Function with return type and with arguments.
   ```
   float area(float len,float wid)
   {
       return(len*wid);
   }
   ```

12. Explain different types of function-calling with an example.

A.

we can call a function in two different ways, based on how we specify the arguments, and these two ways are:

1. Call by Value
2. Call by Reference

Call by Value

Calling a function by value means, we pass the values of the arguments which are stored or copied into the formal parameters of the function. Hence, the original values are unchanged only the parameters inside the function changes.

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

```
void calc(int x);


int main()

{

    int x = 10;

    calc(x);

    // this will print the value of 'x'

    printf("\nvalue of x in main is %d", x);

    return 0;

}


void calc(int x)

{

    // changing the value of 'x'

    x = x + 10 ;

    printf("value of x in calc function is %d ", x);

}
```

In this case, the actual variable x is not changed. This is because we are passing the argument by value, hence a copy of x is passed to the function, which is updated during function execution, and that copied value in the function is destroyed when the function ends(goes out of scope). So the variable x inside the main() function is never changed and hence, still holds a value of 10.

Call by Reference

In call by reference we pass the address(reference) of a variable as argument to any function. When we pass the address of any variable as argument, then the function will have access to our variable, as it now knows where it is stored and hence can easily update its value.

In this case the formal parameter can be taken as a **reference** or a **pointer**(don't worry about pointers, we will soon learn about them), in both the cases they will change the values of the original variable.

```
#include<stdio.h>

void calc(int *p);      // function taking pointer as argument

int main()

{

   int x = 10;

   calc(&x);       // passing address of 'x' as argument

   printf("value of x is %d", x);

   return(0);

}



void calc(int *p)       //receiving the address in a reference pointer variable

{

   /*

      changing the value directly that is

      stored at the address passed
```

```
    */
    *p = *p + 10;
}
```

13. Write about any five string Handling Functions in C language.

A.

Following are some of the useful string handling functions in C.

1. strlen()
2. strcpy()
3. strncpy()
4. strcat()
5. strncat()
6. strcmp()

strlen()
strlen() is used to find length of a string. strlen() function returns the length of a string. It returns integer value.

syntax: int strlen(string name);

strcpy()
strcpy() function is used to copy one string to another. strcpy() function accepts 2 parameters. First parameter is the destination string i.e. the string variable used to copy the string into. Second parameter is the source string i.e. the string variable or string value that needs to be copied to destination string variable.
The Destination_String should be a variable and Source_String can either be a string constant or a variable.

Syntax:

strcpy(Destination_String,Source_String);

strcpy()

strcpy() function is used to copy one string to another. strcpy() function accepts 2 parameters. First parameter is the destination string i.e. the string variable used to copy the string into. Second parameter is the source string i.e. the string variable or string value that needs to be copied to destination string variable.

The Destination_String should be a variable and Source_String can either be a string constant or a variable.

Syntax:

strcpy(Destination_String,Source_String);

strncpy()

strncpy() is used to copy only the left most n characters from source to destination. The Destination_String should be a variable and Source_String can either be a string constant or a variable.

Syntax:

strncpy(Destination_String, Source_String,no_of_characters);

strcat()

strcat() is used to concatenate two strings.

The Destination_String should be a variable and Source_String can either be a string constant or a variable.

Syntax:

strcat(Destination_String, Source_String);

strncat()

strncat() is used to concatenate only the leftmost n characters from source with the destination string.

The Destination_String should be a variable and Source_String can either be a string constant or a variable.

Syntax:

strncat(Destination_String, Source_String,no_of_characters);

strcmp()

strcmp() function is use two compare two strings. strcmp() function does a case sensitive comparison between two strings. The Destination_String and Source_String can either be a string constant or a variable.

Syntax:

int strcmp(string1, string2);

This function returns integer value after comparison.

Value returned is 0 if two strings are equal.

If the first string is alphabetically greater than the second string then, it returns a positive value.

If the first string is alphabetically less than the second string then, it returns a negative value

14. Write a C program to check whether a given string is palindrome or not?

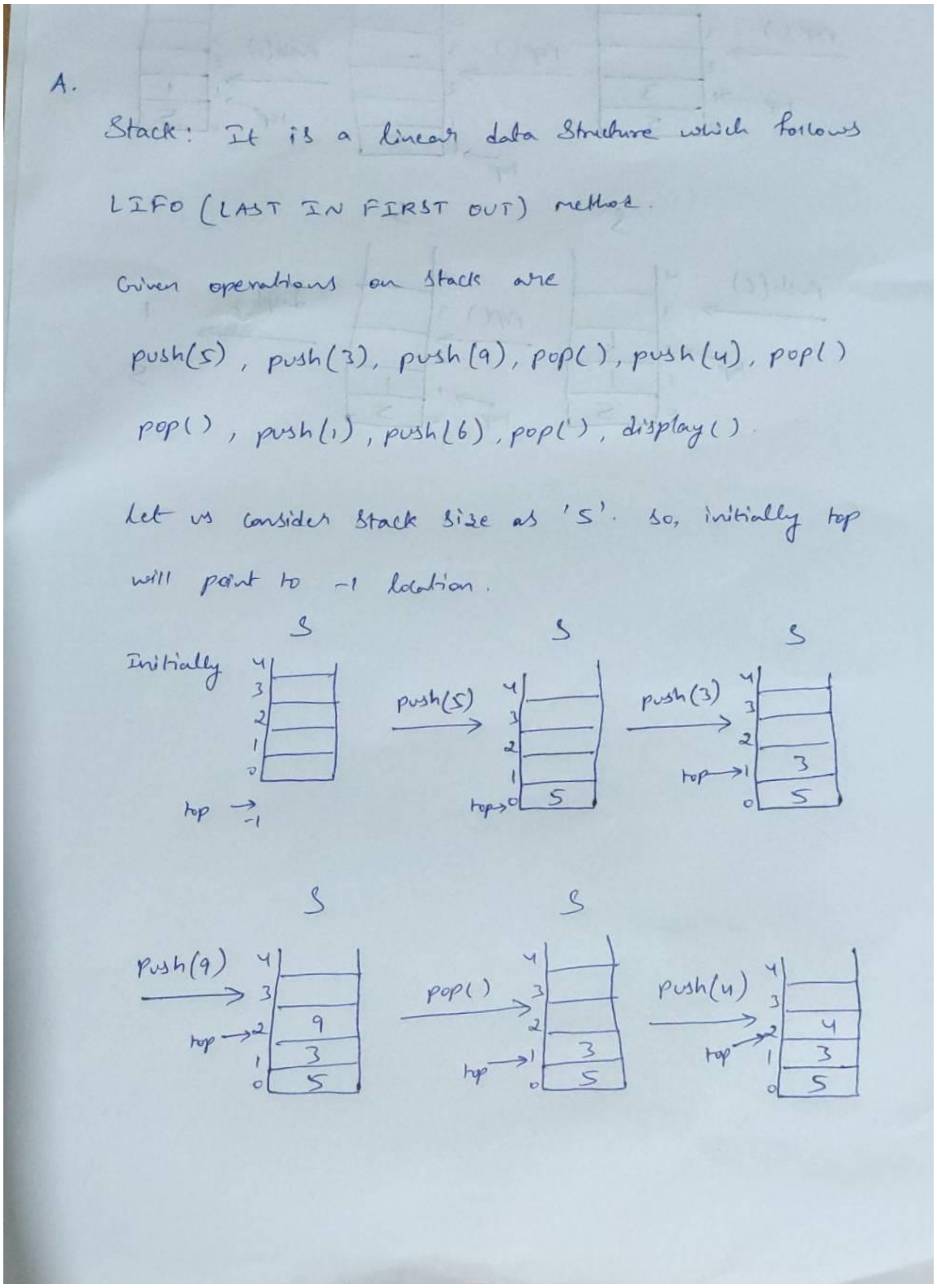
A.

```
#include<stdio.h>
#include<string.h>
void main()
{
        char s1[30],s2[30];
        puts("enter a string:");
        gets(s1);
        strcpy(s2,s1);
        strrev(s2);
        if(strcmp(s1,s2)==0)
        printf("String is a palindrome");
        else
        printf("String is not a palindrome");
}
```

15. Implement the following operations on stack:

push(5), push(3), push(9), pop(), push(4), pop(),pop(), push(1),push(6),pop(), display().

A.



A.

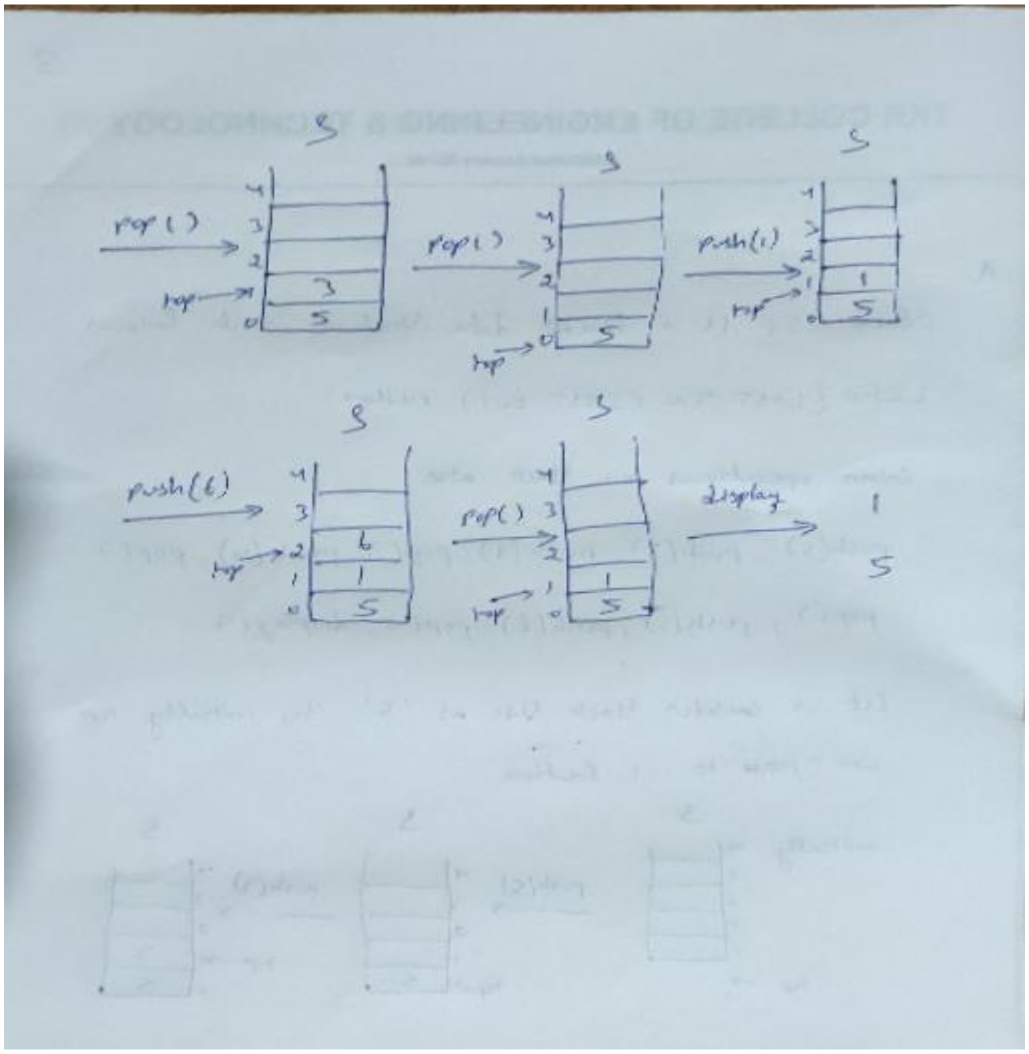
Stack: It is a linear data Structure which follows LIFO (LAST IN FIRST OUT) method.

Given operations on stack are

push(5), push(3), push(9), pop(), push(4), pop() pop(), push(1), push(6), pop(), display().

Let us consider stack size as '5'. So, initially top will point to -1 location.

S
Initially
4
3
2
1
0
top → -1
push(5)
top→0 5
push(3)
top→1 3
5
push(9)
top→2 9
3
5
pop()
top→1 3
5
push(4)
top 4
3
5



16. Define data structure? Briefly explain the operations performed on queue data structure with an example.

A.

Data structure:

It is a particular way of organizing and storing data in a computer is called as data structure .

Queue:

It is a linear data structure in which a data item is inserted at one end and deleted at other end.

The end where we data is inserted is called as “Rear” end and end from where we remove is called “Front” end. Queue uses FIFO technique, First In First Out.

Limitations of Queue:

When an element is removed from linear queue, the location remains unused. Even if the queue is empty, new elements cannot be inserted.

Sequence of values returned by the get operation are as follows:

Initially,

rear=front=-1. (Queue is empty)

J denotes Enqueue→ rear=0

N denotes Enqueue → rear=1

T denotes Enqueue → rear=2

*denotes Dequeue → front=0

U denotes Enqueue → rear=3

*denotes Dequeue → front=1

H denotes Enqueue → rear=4

Y denotes Enqueue → rear=5

D denotes Enqueue → rear=6

*denotes Dequeue → front= 2

*denotes Dequeue → front=3

*denotes Dequeue → front=4

E denotes Enqueue → rear=7

R denotes Enqueue → rear=8

*denotes Dequeue → front=5

*denotes Dequeue → front=6

*denotes Dequeue → front=7

A denotes Enqueue →rear=9

B denotes Enqueue → rear=10

*denotes Dequeue → front= 8

A denotes Enqueue → rear=11

*denotes Dequeue → front=9

*denotes Dequeue → front=10

D denotes Enqueue → rear=12

Now the queue contains, only two elements by the end of these sequence operations.

17. Define file? What are the different types of files? What are the different modes of opening a file?

A.

A file represents a sequence of bytes on the disk where a group of related data is stored.
We have two types of files:--

1.Text file

2. Binary file

We perform different operations on files by using different modes of files.

For Text files

1. read mode(r)

2. write mode(w)
3. append mode(a)

For Binary files

1. read mode(rb)
2. write mode(wb)
3. append mode(ab)

Opening and Closing a file:--

You can open a file either to write data into it or read data from it.

Syntax:--   FILE *file_pointer;

Ex:-   FILE *fp;

Syntax:-   filepointer=fopen("filename","mode");

Ex:--   fp=fopen("sample.c","r");// file has been opened in read mode.

Syntax:-   fclose(Filepointer);

Ex:-   fclose(fp);

```
// Program to demonstrate file

#include<stdio.h>

void main()

{

        FILE *fp;

        fp=fopen("sample.txt","r");

        if(fp==NULL)

        {
```

```
                    printf("FILE DOESNOT EXIST");
        }
        else
        {
                    printf("FILE EXISTS");
        }
        fclose(fp);
}
```

18. Explain about the file positioning functions in C.

A.

File positioning functions

1. ftell ( )

2. rewind ( )

3. fseek ( )

1. ftell ( ) : It returns the current postion of the file pointer

Syntax : int n = ftell (file pointer)

Eg : FILE *fp;

int n;

_____

_____

_____

n = ftell (fp);

Note : ftell ( ) is used for counting the no of characters entered into a file.

2. rewind ( )

It makes the file pointer move to the beginning of the file.

Syntax: rewind (file pointer);

Eg : FILE *fp;

-----

rewind (fp);

n = ftell (fp);

printf (“%d”, n);

o/p: 0 (always).

3. fseek ( )

It is used to make the file pointer point to a particular location in a file.

Syntax: fseek(file pointer,offset,position);

offset :

 The no of positions to be moved while reading or writing.

 If can be either negative (or) positive.

Positive - forward direction.

Negative – backward direction .

position :

 it can have 3 values.

0 – Beginning of the file

1 – Current position

2 – End of the file

19. Write a C program to copy the content of one file to another file.

A.

```
#include<stdio.h>
void main()
{
        FILE *fp,*fp1;
        fp=fopen(“sample.txt”,”r”);
        fp1=fopen(“copy.txt”,”w”);
        if(fp==NULL)
        {
                printf(“FILE NOT EXIST”);
        }
        else
        {
                printf(“FILE EXIST”);
                while((ch=fgetc(fp))!=EOF)
                {
                        fputc(ch,fp1);
                }
}
```

20. Write a C program to display the content of a file on standard output.

A.

```
    #include<stdio.h>
    void main( )
    {
        FILE *fp;.
        char ch;
        fp=fopen(“Hello.txt”,”r”);
        while((ch=getc(fp))!=EOF)
        {
                printf(“%c”,ch);
        }
        fclose(fp);
    }
```

Output:

```
Hi,
We are CSE A Students from TKR College of Engineering and Technology, studying I B.Tech I Sem.
This is a sample text file for the programs of Week 12 to Week 14.
```

21. Write a C program to find factorial of a positive integer.

A.

```
        #include<stdio.h>
        void main()
        {
                int n,I,fact=1;
                printf(“enter a number\n”);
                scanf(“%d”,&n);
                for(i=1;i<=n;i++)
                {
                        fact=fact*n;
                }
        printf(“factotial of %d is %d\n”,n,fact);
        getch();
        }
```

output:

```
enter a number 5
factorial of 5 is 120
```

22. Briefly discuss about storage classes.

A.

Storage classes are classified into 4 kinds

1. auto
2. register
3. static
4. extern

Default of all storage classes is auto.

syntaxes:

auto data_type variable/ function name;

Initial value= Garbage value

register data_type variable/function name;

Initial value= Garbage value

static data_type variable/function name;

Initial value →int=0, float=0.0,string='\0'

extern data_type variable/function name;

Initial value →int=0, float=0.0,string='\0'

Example:

1.auto:

```
#include<stdio.h>
void main()
{
        auto inti=1;
        {
                auto inti=2;
                {
                        auto inti=3;
                        printf("\n%d",i);
                }
                printf("\n%d",i);
        }
        printf("\n%d",i);
}
```

Output:

```
3
2
1
```

2. **register:--**

```
#include<stdio.h>
void main()
{
        register int I;
        for(i=1;i<=5;i++)
        {
                printf("\t%d",i);
        }
}
```

Output:--	1	2	3	4	5

3. **Static:--**

```
#include<stdio.h>
void print();
void main()
{
    print();
    print();
    print();
}
void print()
{
    static inti=1;
    printf("%d\t",i);
    i++;
}
Output:--
1    2    3
```

4. extern:--

```
#include<stdio.h>
int x=20;
intfunc();
void main()
{
        extern int y;
        printf("%d%d",x,y);
}
int y=30;
func()
{
printf("%d%d",x,y);
}
```

23. Briefly discuss about dynamic memory allocation functions?

A.

Memory Allocation Functions:

1. The process of allocating memory while writing the program is known as Static memory allocation.
2. The process of allocating memory during execution is known as Dynamic memory allocation.
3. In C we have 4 library functions, known as Memory management functions.
4. These are useful, when we develop complex application programs.
5. These functions are included in <stdlib.h>
6. 4 memory management functions are as follows:
    a. malloc()
    b. calloc()

   c. realloc()
   d. free()

a. malloc()= allocates block of memory. mainly used with structures.
b. Syntax: ptr=(casttype *)malloc(size);
c. calloc()= allocates multiple blocks of memory, each of which are same size. used with arrays.

   Syntax: ptr=(casttype *)calloc(n,size);
d. realloc()=used to modify the allocated memory space.

   Syntax: ptr=realloc(pointer_name, new size);
e. free()= used to de-allocate the memory allocated by malloc() and calloc().

Example:

```
#include<stdio.h>
void main()
{
        char *p;
        p=malloc(20*sizeof(char));
        if(p==NULL)
        {
                printf("Cannot allocate memory");
        }
        else
        {
                p="computer programming in C";
        }
        printf("Dynamically allocated memory content: %s\n",p);
        p=realloc(p,40*sizeof(char));
        p="Computer Programming in C External Exam";
        printf("Reallocated memory content is :%s",p);
        free(p);
}
```

24. Define structure? Write the syntax for declaring a structure? Briefly explain how to access the members of a structure.

A.

Structures:--

It is a collection of heterogeneous data items or collection of data items of different data types under a single name.

Declaration:--

Syntax:--

```
struct <structure_name>

{

        //Structure members

}Structure variables;

or

struct <structure_name>

{

        //Structure members

};

struct structure variable1, structure variable2----;
```

Initialization:--

In structures members are initialized in different ways.

Ex:-

struct employee

```
{
        char ename[30];
        int empid;
        float esal;
}emp={“xxxyyy”,1600,18850.25};
```

or

```
struct employee
{
        char ename[30];
        int empid;
        float esal;
};
struct employee emp={“xxxyyy”,1600,18850.25};// Member Initialization
```

Accessing of structure members:--

Members of the structure are accessed with the help of structure variables linked to their structure.

By using .(dot) operator members of the structure are accessed.

Ex:--

emp.ename;

→ emp is the structure variable

ename is the Structure member.