LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY

UGC Autonomous

Approved by AICTE | Affiliated to Osmania University

Accredited by NBA | Accredited 'A' grade by NAAC |Certified by ISO.

Estd. 2002

Department of Science and Humanities

Academic Year: 2023-24

Year: I Semester: I

CIE I QUESTION BANK

PROGRAMMING FOR PROBLEM SOLVING

[U23CS101]

[Common to CSE,CSD,CIVIL,MECH]

Prepared by:

Course Coordinator:

Ms. Pooja Chavan,
Assistant Professor,
CSE Department.

Course Faculties:

Mr. Amer Noor Khan, Assistant Professor
Mrs. K Shilpa, Assistant Professor
Mrs. M. Kavitha, Assistant Professor
Mrs. D. Latha, Assistant Professor

Note: A question bank is versatile and flexible FAQs that cover the entire syllabus of a subject. It is used by students and teachers for learning and assessment purposes only.

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY
(UGC Autonomous)
B.E-I SEMESTER

CIE I QUESTION BANK – 2024
Programming for Problem Solving– (U31CS101)
(Common for CSE,CSD,CIVIL,MECH)

Short Answer Questions(SAQs) **Marks - each 2M**

S.NO	UNIT-1	CO	BTL
1.	Differentiate between Algorithms and Flow charts.	CO1	BTL1
2.	Define keywords and variables in C language.	CO1	BTL1
3.	Define flowchart symbols and their usage?	CO1	BTL2
4.	Define Ternary operator with example?	CO1	BTL2
5.	Compare between application software and system software?	CO1	BTL2
6.	Compare between Compiler and Interpreter.	CO1	BTL2
7.	What is a sizeof () operator?	CO1	BTL2
8.	Write any 2 Bitwise operators with example?	CO1	BTL2
9.	Draw a flowchart to display whether given number is positive or negative or zero.	CO1	BTL2
10.	Write an algorithm to find area & perimeter of rectangle.	CO1	BTL2
11.	What is the difference between logical error and syntax error?	CO1	BTL2
12.	Define an algorithm and list its characteristics?	CO1	BTL2
13.	What are Datatypes in C language?	CO1	BTL2
14.	Write any 2 escape sequence characters with example.	CO1	BTL2
15.	Convert the 101010 Binary Number to Decimal Number.	CO1	BTL2
	UNIT-II		
16.	Write the syntax of switch case statement.	CO2	BTL1
17.	Define One-Dimensional array with an example.	CO2	BTL2
18.	Compare between entry controlled and exit controlled loop.	CO2	BTL1
19.	Write a C program to print 1 to 10 using do-while loop?	CO2	BTL2
20.	How string is declared and initialized?	CO2	BTL2
21.	List the different types of decision-making statements.	CO2	BTL2
22.	What is goto statement in C?	CO2	BTL2
23.	Write syntax & example of do-while loop statement.	CO2	BTL2

24.	Define Two-Dimensional array with an example.	CO2	BTL2
25.	Write syntax of else-if ladder and simple if statements in C.	CO2	BTL2
26.	Write about **strcpy()** and **strlen()** functions in C.	CO2	BTL2
27.	Write a C program to print first N natural Numbers using **for** loop?	CO2	BTL2
28.	Write syntax & example of **break** statement.	CO2	BTL2
29.	Write syntax of if-else and nested if-else statements in C.	CO2	BTL2
30.	What are Jumping Statements in C language?	CO2	BTL2
	UNIT-III		
31.	Compare Linear search and Binary Search Algorithms.	CO3	BTL4
32.	Distinguish between Bubble Sort and Selection Sort Algorithms.	CO3	BTL4
33.	List advantages and disadvantages of Bubble sort.	CO3	BTL4
34.	List advantages and disadvantages of Selection sort.	CO3	BTL4
35.	Write Linear Search Algorithm.	CO3	BTL2
36.	List merits and demerits of Linear Search Algorithm.	CO3	BTL4
37.	Define Searching and Sorting algorithms.	CO3	BTL2

Long Answer Questions (LAQs)

S.NO	UNIT-I	MARKS	CO	BTL
1.	Explain the basic structure of C with a program.	6M	CO1	BTL2
2.	Write an Algorithm to check whether given number is even or odd.	6M	CO1	BTL3
3.	Explain the process of creating and running a C program.	6M	CO1	BTL2
4.	Explain the following operators in C. i)Relational ii)Logical iii)Assignment	6M	CO1	BTL2
5.	Write an algorithm to check whether given year is LEAP Year or Not	6M	CO1	BTL3
6.	Draw a flowchart to display largest number among 3 numbers.	6M	CO1	BTL3
7.	Discuss briefly about hardware components of Computer.	6M	CO1	BTL2
8.	Illustrate different types of Tokens in C.	6M	CO1	BTL2
9.	Describe in brief about Data types used in C.	6M	CO1	BTL2
10.	Draw and explain the block diagram of a Computer .	6M	CO1	BTL2
11.	List characteristics of an algorithm and write an algorithm to find area & perimeter of Circle.	6M	CO1	BTL2
12.	Describe in brief about Bitwise Operators in C.	6M	CO1	BTL2
13.	What are the different types of operators used in C and give example	6M	CO1	BTL3
14.	Write an algorithm to display Reverse of a given number?	6M	CO1	BTL3
15.	What are the components of the computer? Explain the functions of each component.	6M	CO1	BTL3
	UNIT-II			
16.	Explain how break and continue statements are used in C.	6M	CO2	BTL4
17.	Explain Iterative statements in C and write their syntax.	6M	CO2	BTL2
18.	Write a C program to perform matrix addition.	6M	CO2	BTL2
19.	Illustrate the following String library functions with syntax. i.strcpy() ii.strcat() iii.strrev() iv.strcmp()	6M	CO2	BTL2
20.	Differentiate while and do-while loops with an example.	6M	CO2	BTL3

21.	Write a C program to display the sum of first N natural numbers using do-while loop.	6M	CO2	BTL3
22.	Explain about nested if, else-if ladder statement with an example.	6M	CO2	BTL2
23.	What is an Array? And explain different types of Arrays.	6M	CO2	BTL2
24.	Explain how to declare and initialize 2-Dimensional array.	6M	CO2	BTL3
25.	Illustrate how to declare and initialize1-Dimensional array.	6M	CO2	BTL3
26.	Describe in brief operator precedence and Associativity with examples?	6M	CO2	BTL2
27.	Define String and Explain any 3 String handling functions used in C?	6M	CO2	BTL2
28.	Write a C program to check whether given number is divisible by 5 and 7.	6M	CO2	BTL3
29.	Write a C program to check if the given string is palindrome or not.	6M	CO2	BTL3
30.	Explain in brief about Switch case statement in C.	6M	CO2	BTL2
UNIT-III				
31	Write Linear search algorithm and Explain how it works with example	6M	CO3	BTL4
32	Write an algorithm to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).	6M	CO3	BTL3
33	Write Bubble sort algorithm and explain how bubble sort works with example.	6M	CO3	BTL4
34	Write Selection sort algorithm and Explain how it works with example.	6M	CO3	BTL4
35	Explain how binary search works to search a number (25) in the given list 45 37 31 28 25 15 7 3.	6M	CO3	BTL4
36	Draw a flowchart to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).	6M	CO3	BTL3
37	Write a C program to convert Binary to Decimal using functions.	6M	CO3	BTL3
38.	Write a C program to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).	6M	CO3	BTL3

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY
(An UGC Autonomous)
B.E-I SEMESTER
CIE I QUESTION BANK with ANSWERS
Programming for Problem Solving
(Common for CSE,CSD,CIVIL,MECH)
ANSWERS for SAQ's

UNIT-1

1. Differentiate between Algorithms and Flow charts.

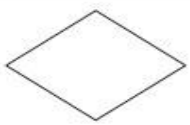
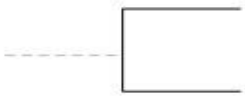
S.No	ALGORITHMS	FLOWCHART
1	An algorithm is a step by step procedure to solve a any problem.	Flowchart is a pictorial representation of an algorithm which takes to process the data.
2	Algorithm does not follow any rules but a meaningful plain text is used.	It utilizes different types of geometrical shapes, symbols, and patterns.
3	Algorithm is the pseudo code for the program.	Flowchart is just graphical representation of a program takes to process data in directions.
4	Difficult to understand and easy to debug errors.	Easy to understand and difficult to debug errors.
5	Difficult to show branching and looping	Easy to show branching and looping.

2. Define keywords, variables in C language.

Keywords--Keywords are predefined, reserved words used in programs that have special meanings to the compiler.Example: int, if, void, else, struct, typedef etc.,

Variables-- A variable is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.Example: Area, Salary, age, name etc.,

3. Define flowchart symbols and their usage?

Flowchart Symbol	Symbol Name	Description
	Terminal (Start or Stop)	Terminals (Oval shapes) are used to represent start and stop of the flowchart.
	Flow Lines or Arrow	Flow lines are used to connect symbols used in flowchart and indicate direction of flow.
	Input / Output	Parallelograms are used to read input data and output or display information
	Process	Rectangles are generally used to represent process. For example, Arithmetic operations, Data movement etc.
	Decision	Diamond shapes are generally used to check any condition or take decision for which there are two answers, they are, yes (true) or no (false).
	Connector	It is used connect or join flow lines.
	Annotation	It is used to provide additional information about another flowchart symbol in the form of comments or remarks.

4. **Define Ternary operator with example?**

We use the ternary operator in C to run one code when the condition is true and another code when the condition is false. It is represented in (**?:**) and also called as Conditional Operator.

Syntax--

Condition**?**value_if_true**:**value_if_false

Example--Output

```
int a = 10, b = 20, c;c value is 10
c = (a < b) ? a : b;
printf("c value is %d", c);
```

5. **Compare between application software and system software?**

S.No	Application Software	System Software
1	Application Software is designed to accomplish tasks for specific purposes of users.	System Software's is mainly designed for managing system resources.
2	Programming of application software is easy.	Programming of system software is comparatively complex.
3	High level languages like JAVA, .NET, HTML etc., are used to develop application softwares.	Low level languages like C/C++, PASCAL, Assembly etc., are used to develop System softwares.
4	**Example:** Notepad, VLC player , Adobe Photoshop , Browsers etc.	**Example:**Operating system, Device drivers , compilers etc.

6. **Compare between Compiler and Interpreter.**

S.No	Compiler	Interpreter
1	The compiler scans the whole program and converts into one low level language.	Interpreter translates the program one statement at a time into low level language.
2	It converts the source code into object code.	It does not convert source code into object code instead it scans it line by line
3	C, C++, C#, etc are programming languages that are compiler-based.	Python, Ruby, Perl, MATLAB, etc are programming languages that are interpreter-based.

7. **What is a sizeof( ) operator?**

sizeof() is a C language operator that returns the size of a variable or data type in bytes.

syntax

sizeof (expression) //Here, expression can be any variable or data type, and sizeof() returns the size of the variable or data type in bytes.

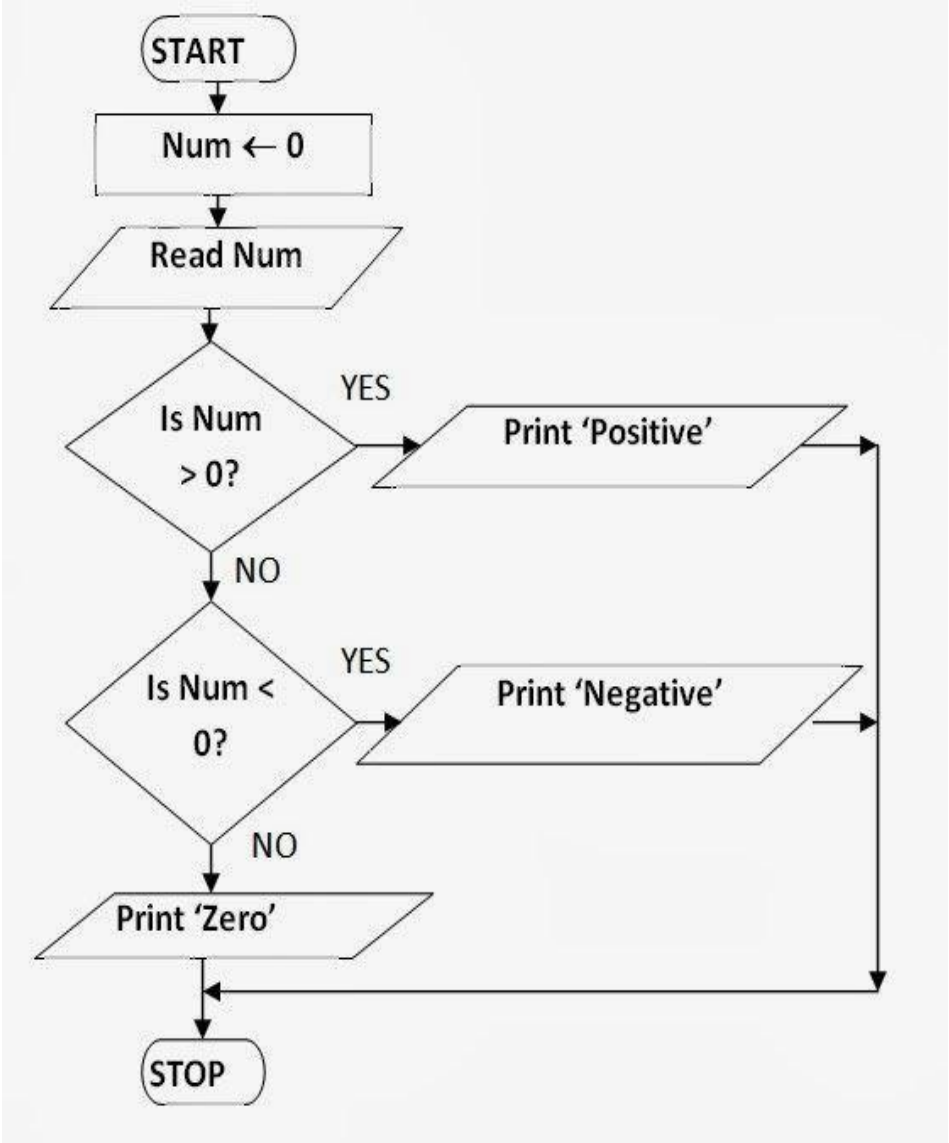
Example

```
int x;
int size = sizeof(x);  // integer variable occupies 2 Bytes so size is 2
```

8. **Write any 2 Bitwise operators with example?**

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12, which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61, which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49, which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61, which is 1100 0011 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

9. **Draw a flowchart to display whether given number is positive or negative or Zero.**



10. **Write an algorithm to find area & perimeter of rectangle.**

START
Step 1 → Take integer variable LENGTH and BREADTH
Step 2 → Assign values to the two variables
Step 3 → Perform Area=LENGTH * BREADTH
Step 4 → Perform Perimeter=2 * (LENGTH + BREADTH)
Step 5 → print Area and Perimeter
STOP

11. **what is the difference between logical error and syntax error?**

S.No	syntax error	logical error
1	Syntax errors are caused by violations of the programming language's syntax rules	logical errors are caused by flaws in the program's logic.
2	These errors are detected by the compiler or interpreter at the time of program compilation or execution	These errors are NOT detected by the compiler or interpreter at the time of program compilation or execution
3	Syntax errors prevent the program from running,	logical errors allow the program to run but produce an incorrect result.
4	**Example:**misspelled keywords, missing or misplaced punctuation, and incorrect use of operators.	**Example:**incorrect calculations, incorrect use of conditional statements, and incorrect loop termination conditions.

12. **Define an algorithm and list its characteristics?**

An algorithm is a step by step procedure for solving any problem or performing a task.
Characteristics of an Algorithm

Finiteness: An algorithm should have finite number of steps and it should end after a finite time.
Input: An algorithm may have many inputs or no inputs at all.
Output: It should result at least one output.
Definiteness: Each step must be clear, well-defined and precise. There should be no any ambiguity.
Effectiveness: Each step must be simple and should take a finite amount of time.

13. **What are Datatypes in C language?**

Data types are used to define the type of data that a variable or function can store or operate on. The C language supports several built-in data types that can be used to declare variables and functions. They are

Basic Data Types **int , float , double , char**

```
int num = 10;          // integer data type
float pi = 3.14;       // floating-point data type
double precision = 0.01;// double precision floating-point data type
char letter = 'A';     // character data type
int arr[5] = {1, 2, 3, 4, 5}; // array data type
```

14. **Write any 2 escape sequence characters with example.**

Escape sequences are special characters used in strings that are not normally printable.
They are represented by a backslash followed by a specific character, and are used to perform certain actions such as inserting newlines, tabs, and other special characters into the string.
1. **Newline character (\n):** This escape sequence is used to insert a new line in the string.
2. Tab character (\t): This escape sequence is used to insert a horizontal tab in the string.

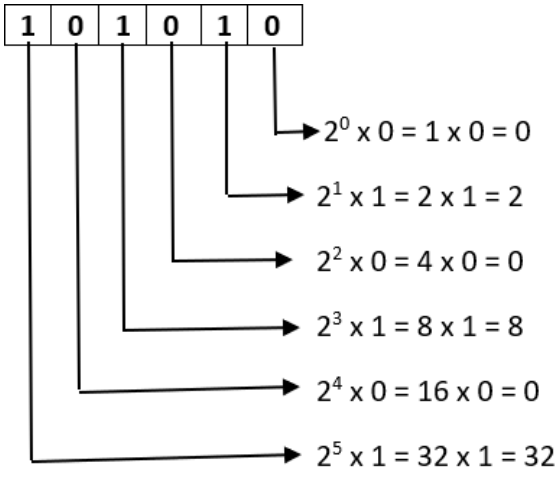
```
#include <stdio.h>
voidmain()
{
printf("Name\tAge\tCity\n");
```

OUTPUT

```
Name    Age     City
John    25      New York
Mary    28      San Francisco
```

```
printf("John\t25\tNew York\n");
printf("Mary\t28\tSan Francisco\n");
}
```

15. **Convert the 101010 Binary Number to Decimal Number.**



Resultant decimal number= 0+2+0+8+0+32 = **42**

UNIT-II

16. **Write Syntax of switch case statement.**

The switch case statement is a control flow statement in programming that is used to execute different code blocks based on the value of a variable or expression.

Syntax---

```
switch (expression)
{
caseconstant1:      // statements/instructions
break;
caseconstant2:      // statements/instructions
break;
   ……
default:// default statements/instructions
}
```

17. **Define One-Dimensional array with an example.**

A One-Dimensional Array in C is a special type of variable that can store multiple values of similar data type such as int, float, etc. at a contagious location in memory.

Elements of an array can be accessed with the same array name by specifying the index number.

Syntax-

data_type **arr_name [arr_size]={value1, value2, value3,…};**

Example-

int **number[5] = {10, 15, 28, 63, 44};** //array number is initialized with elements 0,1,2,3,4

10	15	28	63	44
0	1	2	3	4

←Index Values

18. **Compare between entry controlled and exit controlled loop.**

S.No	Entry controlled Loop	Exit controlled Loop
1	Test condition is checked first, and then loop body will be executed.	Loop body will be executed first, and then condition is checked.
2	If Test condition is false, loop body will not be executed.	If Test condition is false, loop body will be executed once.
3	**for** loop and **while** loop are the examples of Entry Controlled Loop.	**do..while** loop is the example of Exit controlled loop.

4	**for(i=1; i< = 10; i++)**
	{
	sum = sum + i ;
	}
	(Or)
	i=1;
	while
	{
	sum = sum + i;
	i ++;
	}
		i=1;
		do
		{
		sum = sum + i;
		i ++;
		}while(i<=10);

19. **Write a C program to print 1 to 10 using do-while loop?**

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i;
clrscr();
i=1;
do
{
printf("%d \t", i);
i ++;
}while(i<=10);
}
```

OUTPUT

1 2 3 4 5 6 7 8 9 10

20. **How string is declared and initialized?**

Def: A string is **a sequence of characters terminated with a null character '\0'.**

or

The string can be defined as the **one-dimensional array of characters terminated by a null ('\0').**

Syntax---

charstring_variable_name [array_size];

For example:

```
char c[] = "c string"; // declaring and initializing
char name[30] = "AMEER";
char pincode[10] = {'5', '0', '0', '0', '8', '6'};// declaring and initializing
```

21. **List the different types of decision-making statements.**

Decision-making statements in programming languages decide the direction of the flow of program execution. They are

Conditional Statements

- → Simple if statement
- → if..else statements
- → else-if ladder
- → nested if statements
- → switch statements

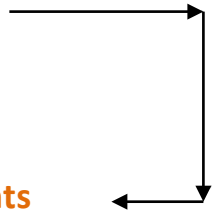
22. **What is goto statement in C?**

goto: It is used **after the** **normal sequence of program execution by transferring the control to some other part of program.**

Syntax

Forward jump

```
gotoLabel ;
        -----
        -----
        -----
Label :statements
```



PROGRAM-2

```
#include<stdio.h>
void main( )
{
printf("\n Hello");
gotoLevel-1;
printf("LORDS College");
printf("CSE branch");
Level-1:
printf("\n This is Goto Statement");
printf("\n End of Go To");
}
```

OUTPUT:

```
Hello
This is Goto Statement
End of Go To
```

23. **Write syntax & example of do-while loop statement.**

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

Syntax---

```
initialization;
do
{
//statements to be executed repeatedly;
increment/decrement;
} while( condition-checking/expression ) ;
```

Example:-

```
i=1;
do
 {
sum = sum + i;
i ++;
}while(i<=10);
```

24. **Define Two-Dimensional array with an example.**

A two-dimensional array is a collection of elements organized in a grid-like format with rows and columns.

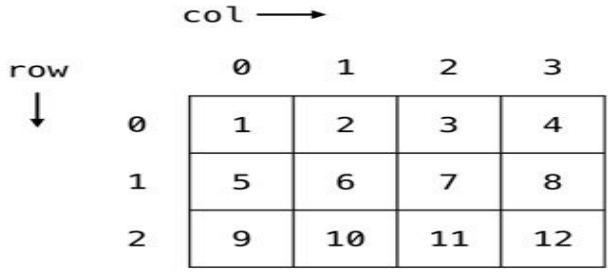
Each element in the two-dimensional array is identified by its row and column index.

syntax

data_type array_name[rows][columns];

Example:

```
int num[3][4] = {
  {1, 2,  3,  4},
  {5, 6,  7,  8},
  {9, 10, 11, 12}
};
```



25. **Write syntax of else-if ladder and simple if statements in C.**

else-if Ladder statement

Syntax---

```
if(condition1)
{
        //code to be executed if condition1 is true
}
else if(condition2)
{
        //code to be executed if condition2 is true
}
else if(condition3)
{
        //code to be executed if condition3 is true
}
else
{
        //code to be executed if all the conditions are false
}
```

Simple if statement----

Syntax---

```
if(expression)
{
        //code to be executed
}
```

26. **Write about strcpy() and strlen() functions in C.**

The strcpy() and strlen() functions are important string manipulation functions in the C.

The strcpy() function is used to copy the contents of one string to another string.

strcpy() ------strcpy(string1, string2)---- Copies string2 value into string1

The strlen() function is used to find the length of a string.

strlen()--------strlen(string1) ---returns total number of characters(int)in string1

27. **Write a C program to print first N natural Numbers using for loop?**

```
void  main()
{
   int n, i;
printf("Enter the value of N: ");
scanf("%d", &n);

   for (i = 1; i<= n; i++)
{
printf("%d \t ", i);
   }

}
```

OUTPUT

```
Enter the value of N: 10
    1    2    3    4    5    6    7    8    9    10
```

28. **Write syntax & example of break statement.**

The break statement **ends the loop immediately** when it is encountered.

Syntax—

```
any loop                    switch(exp)
{                           {
   statements;                 case value: statements;
   break;                                  break;
}                              default:
                             }
```

```
void main()
{
    int i;
 for (i=1; i<=5; i++)
   {
printf ("IT-A");
   break;  //this will skip/break from loop
     printf ("CSE");
}
printf("\n End of Main");
}
Output:
          IT-A
End of Main
```

29. **Write syntax of if-else and nested if-else statements in C.**

if else statement----

Syntax-----

```
if(expression)
{
      //code to be executed if condition is true
}
else
{
       //code to be executed if condition is false
}
```

NESTED if else statement----

Syntax---

```
if(condition1)
{
          //code to be executed if condition1 is true

           if(condition2)
          {
                    //code to be executed if condition2 is true
          }

}
```

```
else
{
        if(condition3)
        {
                //code to be executed if condition is true
        }
}
```

30. **What are Jumping Statements in C language?**

Jumping statements is also known as Unconditional branching which is when the programmer forces the execution of a program to jump to another part of the program.
C programming language allows jumping from one statement to another. It also supports **break, continue, return** and **goto** jump statements.

- The **break** statement **ends the loop immediately** when it is encountered.
- **goto** is used **after the normal sequence of program execution by transferring the control to some other part of program**.
- **continue** statement **skips some lines of code inside the loop and continues**with the next iteration.

Unit-III

31. **Compare Linear and Binary Search algorithms.**

	LINEAR SEARCH	BINARY SEARCH
1.	Data can be any order.	Data should be in sorted order.
2.	Multi-Dimensional array also can be used.	Only Single dimensional array is used.
3.	Not an efficient method to be used if there is a large list.	Efficient for large list also.
4.	Traversing is slow.	Traversing is Fast.
5.	List is searched sequentially	List are partitioned into two parts and compared with key value.
6.	Time Complexity--O(N)	Time Complexity--O(log N)

32. **Distinguish between Bubble Sort and Selection Sort Algorithms.**

	BUBBLE SORT	SELECTION SORT
1.	Adjacent element is compared and swapped	finding the minimum element and then inserting it in its correct position by swapping.
2.	It uses an exchanging method.	It uses a selection method.
3.	It is slower than the selection sort as a greater number of comparisons is required.	It is faster than the bubble sort as a lesser number of comparisons is required.
4.	Inefficient as it is basic sorting technique.	Improved efficiency as compared to bubble sort.
5.	The time complexities in best case and worst case are $O(n)$ and $O(n^2)$ respectively.	The time complexity in both best and worst cases is $O(n^2)$

33. **List advantages and disadvantages of Bubble sort.**

Advantages

- ❖ Bubble sort is easy to understand and implement
- ❖ Bubble sort is an in-place sorting algorithm, meaning that it does not require any extra memory to sort the input array.
- ❖ Bubble sort is an adaptive sorting algorithm
- ❖ Bubble sort is a stable sorting algorithm

Disadvantages

- → The time complexity of selection sort is $O(n^2)$, which means that its performance decreases rapidly as the size of the input array increases.
- → Bubble sort is not efficient for large datasets due to its time complexity.
- → Bubble sort is not suitable for sorting complex data structures like trees and graphs.
- → Bubble sort is not commonly used in practice

34. **List advantages and disadvantages of Selection sort.**

Advantages

- ❖ The implementation of selection sort is easy to understand.
- ❖ Selection sort is an in-place sorting algorithm, meaning that it does not require any extra memory to sort the input array.
- ❖ Works well with small datasets.

Disadvantages

- → The time complexity of selection sort is $O(n^2)$, which means that its performance decreases rapidly as the size of the input array increases.
- → Selection sort is not a stable sorting algorithm,
- → Inefficient for large datasets.
- → High number of comparisons.
- → Selection sort is not an adaptive sorting algorithm,

35. **Write Linear Search Algorithm.**

Linear Search (Array A, Value x)

Step 1: Set i to 1
Step 2: if i> n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit

36. **List merits and demerits of Linear Search Algorithm.**

Merits of Linear Search Algorithm:

- ✓ Linear search is a very simple algorithm to implement.
- ✓ Versatility: Linear search can be used in lists, arrays, and trees.
- ✓ Linear search can be performed on an unsorted list
- ✓ Linear search has a very memory efficiency.

Demerits of Linear Search Algorithm:

- ❖ Linear search has a time complexity of O(n), where n is the size of the data structure being searched.
- ❖ Inefficient for sorted data.
- ❖ Linear search must examine every element in the data set, even if the target element is found early in the search.
- ❖ Inefficient for complex data structures.

37. **Define Searching and Sorting algorithms.**

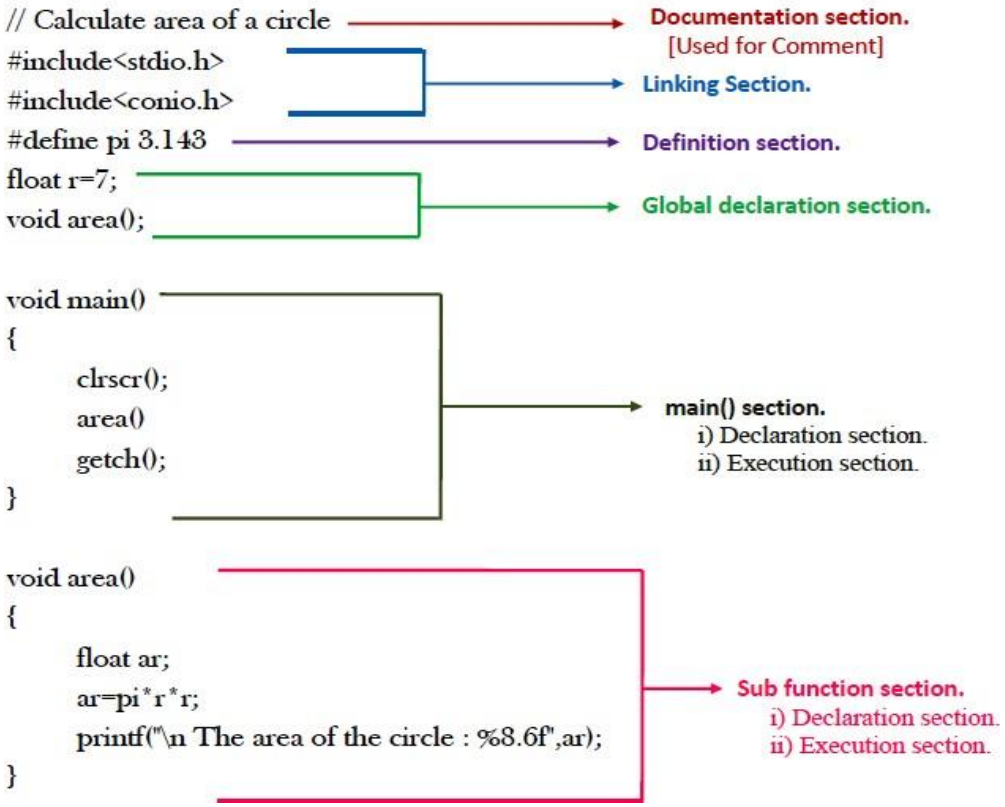
Searching algorithms are used to find a specific element (or set of elements) in a given data structure, such as an array, list, or tree. The goal of a searching algorithm is to determine whether a given element is present in the data structure and, if so, to locate its position.

Sorting algorithms are used to arrange a collection of elements in a specific order, such as ascending or descending. The goal of a sorting algorithm is to organize the data in a way that makes it easier to search, process, or analyze.

LONG ANSWER QUESTIONS
UNIT I

1 **Explain the basic structure of C with a program.**

Basic Structure of C Program



2. **Write an Algorithm to check whether given number is even or odd.**

Algorithm:

It is the set of instructions that is to be executed sequentially to perform a specific task successful.

START

Step 1 → Take integer variable A Step 2

→ Assign value to the variable

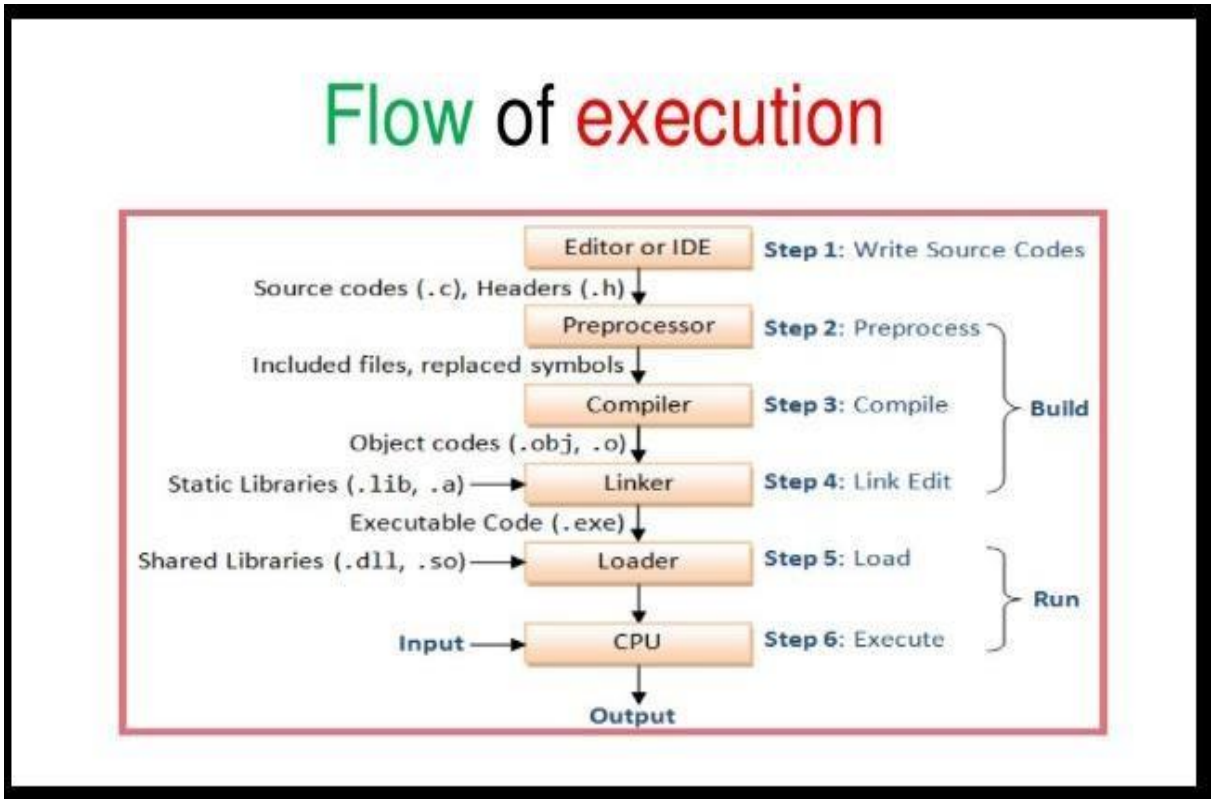
Step 3 → Perform A modulo 2 and check result if output is 0

Step 4 → If true print A is even

Step 5 → If false print A is odd

STOP

3. **Explain the process of creating and running a C program.**



IDE: An integrated development environment (IDE) is software for building applications that combines common developer tools into a single graphical user interface (GUI).
Example : Turbo C/C++, Dev C++, Visual Studio,etc.,

Preprocessor Directives: a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation. We'll refer to the C Preprocessor as CPP. All preprocessor commands begin with a hash symbol (#). Eg: #define,#typedef,#include

Compiler: compiler, computer software that translates (compiles) source code written in a high-level language (e.g., C++) into a set of machine-language instructions that can be understood by a digital computer's CPU. Compilers are very large programs, with error-checking and other abilities.

LINKER:
A linker is special program that combines the object files, generated by compiler/assembler, and other pieces of codes to originate an executable file have. exe extension.
In the object file, linker searches and append all libraries needed for execution of file.
It also merges two or more separate object programs and establishes link among them.

Loader: loader is special program that takes input of object code from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program.

4. **Explain the following operators in C.**
i)Relational ii)Logical iii)Assignment

i)Relational(>,<,>=,<=,==,!=)
ii)Logical(&&,||,!)
iii)Assignment(+=,-=,*=,/=,%=)

Relational Operators in 'C'

==	Equal to	a==b returns 1 if a and b are the same
>	Greater than	a>b returns 1 if a is larger than b
<	Less than	a<b returns 1 if a is smaller than b
>=	Greater than or equal to	a>=b returns 1 if a is larger than or equal to b
<=	Less than or equal to	a<=b returns 1 if a is smaller than or equal to b
!=	Not equal to	a!=b returns 1 if a and b not the same

Logical Operators

- Logical refers to the ways relationships can be connected.
- Expressions that use logical operators return 0 for false & 1 for true.

Operator	Meaning
&&	AND
||	OR
!	NOT

- The truth table for logical operators is shown below:

P	Q	P && Q	P || Q	!P
0	0	0	0	1
0	1	0	1	1
1	1	1	1	0
1	0	0	1	0

Assignment Operators

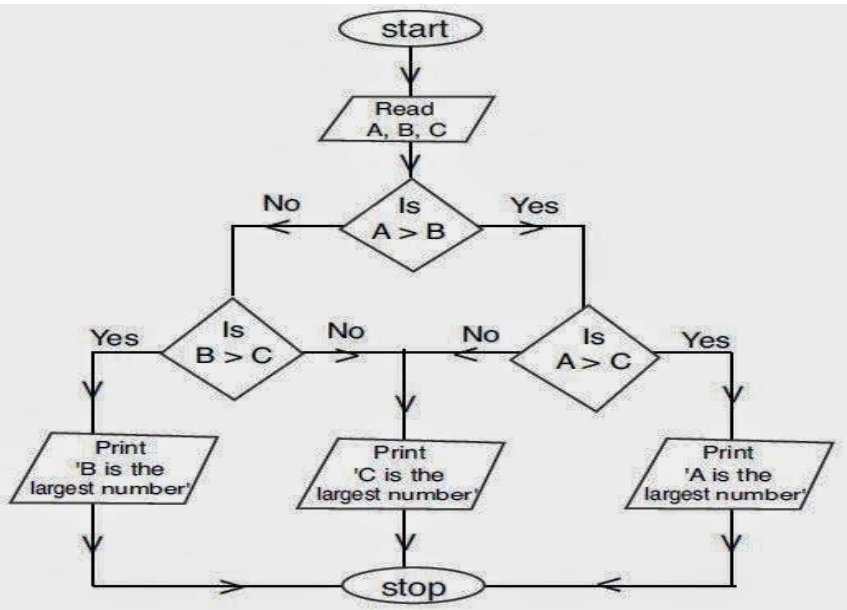
Operator	Example	Equivalent Expression (m=15)	Result
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=	m = m/10	1
%=	m %=10	m = m%10	5

5. **Write an algorithm to check whether given year is LEAP Year or Not.**

START
Step 1 → Take integer variable year
Step 2 → Assign value to the variable
Step 3 → Check if year is divisible by 4 but not 100, DISPLAY "leap year"
Step 4 → Check if year is divisible by 400, DISPLAY "leap year"
Step 5 → Otherwise, DISPLAY "not leap year"
STOP

6.

Draw a flowchart to display largest number among 3 numbers.



7.

Discuss briefly about hardware components of Computer.

Hardware Components of Computer

- Motherboard
- Input Devices
- Output Devices
- Central Processing Unit (CPU)
- Storage Devices

Motherboard---

A motherboard is a circuit board through which all the different components of a computer communications and it keeps everything together. The input and output devices are plugged into the motherboard for function.

Input devices

All the data received by the computer goes through the input unit. The input unit has different devices. Like a *mouse, keyboard, scanner, etc.*

Output devices

All the information sent to the computer once processed is received by the user through the output unit. Devices like *printers, monitors, projector,* etc.

CPU – Central Processing Unit

CPU is the brain of the computer which controls all tasks.
CPU comprises of two units, namely – *ALU (Arithmetic Logic Unit) and CU (Control Unit).*

Storage Devices

Storage Devices are the place files and programs are stored and processed the data according to the request of User. They are two types.

- **Primary memory** – This type of memory cannot store a vast amount of data. The data stored in this is temporary. Eg: *RAM,ROM etc.*

Secondary memory – This type of memory stores a large amount of data. The data stored in this is permanent. Eg: *Hard Disk, Pen drive etc.*

8.

Illustrate different types of Tokens in C.

TOKEN is the smallest unit in a 'C' program. C Token is divided into six different types, viz,

1. Keywords,
2. Identifiers.
3. Strings,
4. Constants,
5. Special Symbols
6. Operators.

Strings

Strings in C is an array of characters having null character '\0' at the end of the string. Strings in C are enclosed in double-quotes("") and Characters are enclosed in single quotes(").

Example:

```
char a[10]={'1','2','3'};
char a[]="Amardeep";
char a[10]="Paramesh";
```

Keywords

Keywords are predefined or reserved keywords used to expose the behaviour of the data. There are 32 keywords in C. Each keyword has its functionality to do.

Example: if , int , float , void, char , struct , for, while, etc.

Identifiers.

- Each program elements in a C program are given a name called identifiers.
- Names given to identify Variables, functions and arrays are examples for identifiers. eg. x is a name given to integer variable in above program.

Constants.

- C Constants are also like normal variables. But, only difference is, their values cannot be modified by the program once they are defined.
- Constants refer to fixed values. They are also called as literals

Example

1. const int **PI3.142**;

2. #define**PI3.142**

Operators

Operators are symbols that help in performing operations of mathematical and logical nature.

Arithmetic Operators (+,-,*,%,/)

1. Relational Operators(>,<,>=,<=,==,!=)
2. Logical Operators(&&,||,!)
3. Bitwise Operators(&,|,^,~,<<,>>)
4. Assignment Operators(=,+=,-=,*=,%=,/=)
5. Increment/Decrement Operators(++,--)
6. Ternary Conditional Operators(?:)
7. Special Operators(&,*,sizeof())

Special Symbols.

The special symbols have some special meaning and they cannot be used for other purposes.

Some of the special symbols that are used in C programming are as follows **[] () {}, ; * = # etc.,**

Parentheses()– These special symbols are used for function calls and function parameters.

Braces{} – Opening and closing of curly braces indicates the start and end of a block of code which contains more than one executable statement.

Pre-processor(#) – The pre-processor called as a macro processor is used by the compiler to transform your program before the actual compilation starts.

9.

Describe in brief about Data types used in C.

Data type specify the type of value stored in a variable

C support following datatypes:

Primary Data type	int, float, char, void
Derived Data Type	Array, Pointer.
User Defined Data type	structure, union, enum, typedef

Integer data type:

Ex: int a=10;

Here a is an integer variable that will store value 10. Integer will need 2 bytes/4 bytes memory storage.

Float data type:

Ex: float b=2.2;

Here b is a float variable that will store value 2.2. float will need 4 bytes of memory storage

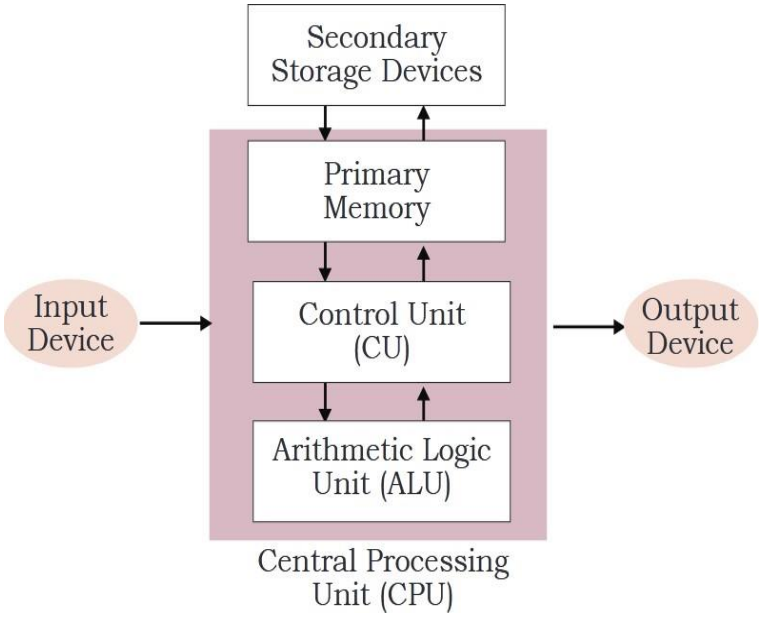
Character data type:

Ex: char ch='a';

ch is a character variable that will store any alpha numeric value enclosed in single quote. The length need to be exact 1 character will need 1 byte of memory storage space.

10.

Draw and explain the block diagram of a Computer .



Input devices
All the data received by the computer goes through the input unit. The input unit has different devices. Like a mouse, keyboard, scanner, etc.

Output devices
All the information sent to the computer once processed is received by the user through the output unit. Devices like printers, monitors, projector, etc.

CPU – Central Processing Unit
CPU is the brain of the computer which controls all tasks.
CPU comprises of two units, namely – ALU (Arithmetic Logic Unit) and CU (Control Unit).

Primary memory – This type of memory cannot store a vast amount of data. The data stored in this is temporary.**Eg:** RAM,ROM etc.

Secondary memory – This type of memory stores a large amount of data. The data stored in this is permanent**. Eg:** Hard Disk, Pen drive etc.

11.

List characteristics of an algorithm and write an algorithm to find area & perimeter of Circle.

Characteristics of an Algorithm

Finiteness: An algorithm should have finite number of steps and it should end after a finite time.
Input: An algorithm may have many inputs or no inputs at all.
Output: It should result at least one output.
Definiteness: Each step must be clear, well-defined and precise. There should be no any ambiguity.
Effectiveness: Each step must be simple and should take a finite amount of time.

Algorithm to find area & perimeter of Circle

Step 1: Start
Step 2: Take Input variables as **radius** and **pi=3.142**accept.
Step 3: Take other variables as **AREA** and **PERIMETER**to store the result.
Step 4: **AREA** =**pi*radius*radius**
Step 5: **PERIMETER**= 2***pi*radius**
Step 6: print **AREA** and **PERIMETER**
Step 7: stop

12.

Describe in brief about Bitwise Operators in C.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12, which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61, which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49, which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61, which is 1100 0011 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

13.

What are the different types of operators used in C and give example

C divides the operators into the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Bitwise operators

Arithmetic Operators

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1

Relational Operators in 'C'

==	Equal to	a==b returns 1 if a and b are the same
>	Greater than	a>b returns 1 if a is larger than b
<	Less than	a<b returns 1 if a is smaller than b
>=	Greater than or equal to	a>=b returns 1 if a is larger than or equal to b
<=	Less than or equal to	a<=b returns 1 if a is smaller than or equal to b
!=	Not equal to	a!=b returns 1 if a and b not the same

Logical Operators

- Logical refers to the ways relationships can be connected.
- Expressions that use logical operators return 0 for false & 1 for true.

Operator	Meaning
&&	AND
||	OR
!	NOT

- The truth table for logical operators is shown below:

P	Q	P && Q	P || Q	!P
0	0	0	0	1
0	1	0	1	1
1	1	1	1	0
1	0	0	1	0

Assignment Operators

Operator	Example	Equivalent Expression (m=15)	Result
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=	m = m/10	1
%=	m %=10	m = m%10	5

14.

Write an algorithm to display Reverse of a given number?

Algorithm
step 1: Start
step 2: Intilize reverse=0.
step 3: Read digit
step 4: Check whether digit>0 then go to step 5
else go to step 9
step 5: reverse =reverse *10
step 6: reverse=reverse+digit%10
step 7: digit =digit/10
step 8: Go to step 4
step 9: Print reverse
step 10: Stop

15.

What are the components of the computer? Explain the functions of each component.

Hardware Components of Computer

- Motherboard
- Input Devices
- Output Devices
- Central Processing Unit (CPU)
- Storage Devices

Motherboard---

A motherboard is a circuit board through which all the different components of a computer communications and it keeps everything together. The input and output devices are plugged into the motherboard for function.

Input devices

All the data received by the computer goes through the input unit. The input unit has different devices. Like a *mouse, keyboard, scanner, etc*.

Output devices

All the information sent to the computer once processed is received by the user through the output unit. Devices like *printers, monitors, projector,* etc.

CPU – Central Processing Unit

CPU is the brain of the computer which controls all tasks.
CPU comprises of two units, namely – *ALU (Arithmetic Logic Unit) and CU (Control Unit).*

Storage Devices

Storage Devices are the place files and programs are stored and processed the data according to the request of User. They are two types.

Primary memory – This type of memory cannot store a vast amount of data. The data stored in this is temporary. Eg: *RAM,ROM etc.*

Secondary memory – This type of memory stores a large amount of data. The data stored in this is permanent. Eg: *Hard Disk, Pen drive etc.*

16. **Explain how break and continue statements are used in C.**

Unconditional branching is when the programmer forces the execution of a program to jump to another part of the program.

C language allows jumping from one statement to another using

→ **break**
→ **continue**
→ **go to** and
→ **return** jump statements.

break Statement

The break statement **ends the loop immediately** when it is encountered.

Syntax—

```
any loop                    switch(exp)
{                           {
   statements;                 case value: statements;
   break;                            break;
}                                 default:
                                  }
```

```
void main()
{
    int i;
 for (i=1; i<=5; i++)
   {
printf ("IT-A");
   break;  //this will skip/break from loop
     printf ("CSE");
}
printf("\n End of Main");
}
Output:
        IT-A
End of Main
```

2. continue Statement

The continue statement in C language is used **to bring the program control to the beginning of the loop**.

The continue statement **skips some lines of code inside the loop and continues** with the next iteration.

Syntax—

```
any loop
{
   statements;
continue;
}
```

```
void main()
{
    int i;
 for (i=1; i<=5; i++)
   {
printf ("IT-A \t");
  continue ;
//skips the next  Statement & continues with
//the next iteration.
     printf ("CSE");
}
printf("\n End of Main");
}
Output:
      IT-A   IT-A   IT-A   IT-A   IT-A
End of Main
```

17. **Explain Iterative statements in C and write their syntax.**

Iterative Statements----

Repetition/Execution of Same or old instruction/statement /block multiple times in a program we called it as LOOPS or Iterative statements.

C programming provides us

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

1. while Loop

A **while** loop in C **repeatedly executes** a target statement as long as a given **condition is true.**

Syntax---

initialization;

```
while( condition-checking/expression )
{
//statements to be executed repeatedly;
increment/decrement;
}
```

2. do_while Loop

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

Syntax---

```
initialization;
do
{
//statements to be executed repeatedly;
increment/decrement;
} while( condition-checking/expression ) ;
```

3. for Loop

A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax---

```
for(initialization; condition-checking; increment/decrement )
{
//set of statements to be executed repeatedly;
}
```

18.

Write a C program to perform matrix addition.

```
#include <stdio.h>
int main()
{
  int rows, columns;
  int matrix1[rows][columns], matrix2[rows][columns], result[rows][columns];
printf("Enter the number of rows and columns for the matrices:\n");
scanf("%d%d", &rows, &columns);
printf("Enter elements for first matrix:\n");
  for (int i = 0; i< rows; i++)
{
    for (int j = 0; j < columns; j++)
{
scanf("%d", &matrix1[i][j]);
    }
  }
printf("Enter elements for second matrix:\n");
  for (int i = 0; i< rows; i++)
{
    for (int j = 0; j < columns; j++)
{
scanf("%d", &matrix2[i][j]);
    }
  }
  for (int i = 0; i< rows; i++)
{
    for (int j = 0; j < columns; j++)
{
```

```
        result[i][j] = matrix1[i][j] + matrix2[i][j];
      }

}   printf("The sum of the two matrices is:\n");
   for (int i = 0; i< rows; i++)
{
     for (int j = 0; j < columns; j++)
{
printf("%d\t", result[i][j]);
      }
printf("\n");
   }
}
```

OUTPUT

Enter the number of rows and columns for the matrices:
2
2
Enter elements for first matrix:
2 4
5 7
Enter elements for second matrix:
5 8
8 9
The sum of the two matrices is:
32 36
32 36

19. **Illustrate the following String library functions with syntax.**
i.strcpy() ii.strcat() iii.strrev() iv.strcmp()

i. strcpy()-- copying source string into destination string.
syntax-strcpy(Destination string, Source String);

ii. strcat() -- It combines two strings and store the result in Destination String.
syntax-strcat (Destination String, Source string);

iii. strrev()--The function is used for reversing a string. reversed string will be stored in the same string.
syntax-strrev (string)

iv. strcmp()--This function compares 2 strings if the two strings are same then it returns 0.
syntax--int strcmp (string1, string2);

20. **Differentiate while and do-while loops with an example.**

While Loop	Do-While Loop

Syntax
while(exp)
{
Statement1;
Statement2;
.
.
Statementn;
}
	Syntax
	do
	{
	Statement1;
	Statement2;
	.
	.
	Statementn;
	} while(exp);

It is a top testing or entry controlled loop since the condition is checked in the beginning of the loop.	It is a bottom testing or exit controlled loop since the condition is checked at the bottom of the loop.
As long as the exp is evaluated to true, the body of the while loop is executed. If the exp is evaluated to false in the beginning of the loop, then the statements within the body of the while loop will not be executed even once.	The body of the do-while also will be executed as long as exp is evaluated to true. But since the condition is checked at the bottom of the loop, the body of the do-while will be executed at least once.

Example:
i=0, sum=0;
while(i<=n)
{
sum=sum+i;
i=i+1;
}
	Example:
	i=0,sum=0;
	do
	{
	sum=sum+i;
	i=i+1;
	} while(i<=n);

21.

Write a C program to display the sum of first N natural numbers using do-while loop.

```
#include <stdio.h>
void main()
{
   int num, i, sum = 0;
  printf(" Enter a positive number: ");
  scanf("%d", &num);
 i = 0;
 do
 {
      sum = sum +i;
       i++;
  }while(i<=num);
   printf("\n Sum of the first %d number is: %d", num, sum);
}
```

OUTPUT:

```
Enter a positive number: 5
Sum of the first 5number is: 15
```

22.

Explain about nested if, else-if ladder statement with an example.

else-if ladder statement

The else..if ladder statement is useful when you need to check multiple conditions within the program. The if-else-if ladder statement is an extension to the if-else statement.

Syntax	Example

if(condition1)
{
 	
// executed if condition1 is true
 	
}
else if(condition2)
{
//executed if condition2 is true
}
else if(condition3)
{
//executed if condition3 is true
}
else
{
//executed if all the conditions are false
}
	int number=30;
	if(number = =10)
	{
	printf("number is equals to 10");
	}
	else if(number = = 50)
	{
	printf("number is equal to 50");
	}
	else
	{
	printf("number is INVALID ");
	}
	
	Output
	number is INVALID

Nested if-else Statement

We can combine the if statement or if-else statement inside the if block, or else block or both if and else. This is called nesting of if-else

Syntax	Example

if(condition1)
{
if(condition2)
{
//true block
}
else
	int a= 23, b = 40;
	if (a>= b)
	{
	if(a == b)
	{
	printf("Both are same");
	}

```
        {
          // false block
}
}
else
{
//false block of condition 1;
}
next statement;
```

```
          else
          {
          printf(" a is Biggest");
          }
}
else
{
printf("b is Biggest");
}
Output:
b is Biggest
```

23.

What is an Array? And explain different types of Arrays.

An Array is a collection of elements of the same data type, identified by a single variable name and an index or subscript that is used to access each element.

Arrays are an essential data structure in programming and are used to store and manipulate a large number of values efficiently. There are different types of arrays, including:

One-dimensional Array: Also known as a linear array, it is the most common type of array. It is a list of elements that can be accessed using a single index or subscript. One-dimensional arrays can be used to represent a list of items or a sequence of data.

Syntax: **datatype array_name [size];**

datatype: It denotes the type of the elements in the array.

array_name: Name of the array. It must be a valid identifier.

size: Number of data elements an array can hold/store.

Declarations and Initialization of an Array Variables:

1. int **x[3]={20,30,5};**

2. int **arr[] = { 10, 20, 30, 40 };**

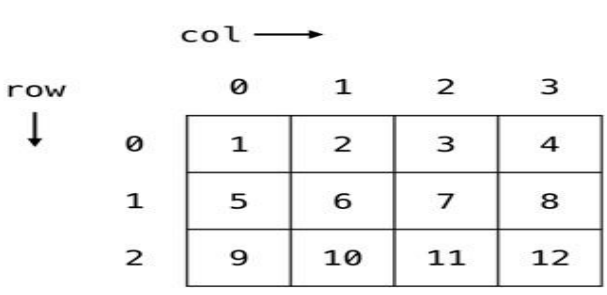
Two-dimensional Array: A multi-dimensional array is an array with more than one index or subscript. It can be thought of as a table with rows and columns.

syntax

data_type array_name[rows][columns];

Example:

```
int num[3][4] = {
  {1, 2,  3,  4},
  {5, 6,  7,  8},
  {9, 10, 11, 12}
};
```



24.

Explain how to declare and initialize 2-Dimensional array.

Two-dimensional Array: A multi-dimensional array is an array with more than one index or subscript. It can be thought of as a table with rows and columns.

syntax

data_type array_name[rows][columns];

```
#include<stdio.h>
void main()
{
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++)
{
   for(j=0;j<3;j++)
{
   printf("arr[%d] [%d] = %d \n", i, j,arr[i][j]);
}//end of j
}//end of i
printf("\n End of Loop");
}
```

OUTPUT:

```
arr[0] [0] = 1
arr[0] [1] = 2
arr[0] [2] = 3

arr[1] [0] = 2
arr[1] [1] = 3
arr[1] [2] = 4

arr[2] [0] = 3
arr[2] [1] = 4
arr[2] [2] = 5

arr[3] [0] = 4
arr[3] [1] = 5
arr[3] [2] = 6
End of Loop
```

25.

Illustrate how to declare and initialize 1-Dimensional array.

Def: An array is defined as the **collection of similar type of data items** stored at **contiguous memory locations**.

Syntax: **datatype array_name[size];**

datatype: It denotes the type of the elements in the array.

array_name: Name of the array. It must be a valid identifier.

size: Number of data elements an array can hold/store.

Declarations and Initialization of an Array Variables:

1.int **x[3]={20,30,5};**

2. int **arr[] = { 10, 20, 30, 40 };**

// Compiler creates an array of size 4. And above is same as "int arr [4] = {10, 20, 30, 40}"

3.int **arr[7] = { 10, 20, 30, 40 };**

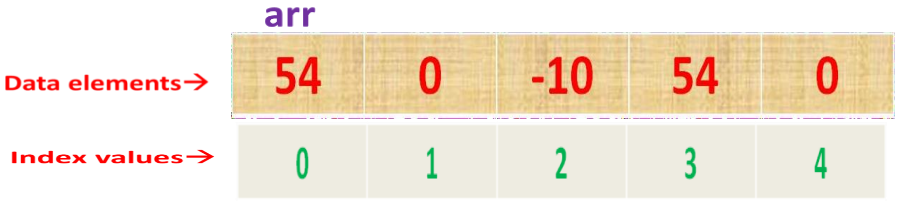
// Compiler creates an array of size 7, initializes first 4 elements as specified by user and rest 3 elements as 0 or Garbage value. and is same as "int arr[] = {10, 20, 30, 40, 0, 0,0 };"

4. int **arr[5];**

arr[0] = 54;

arr[2] = -10;

arr[3] = arr[0];



26.

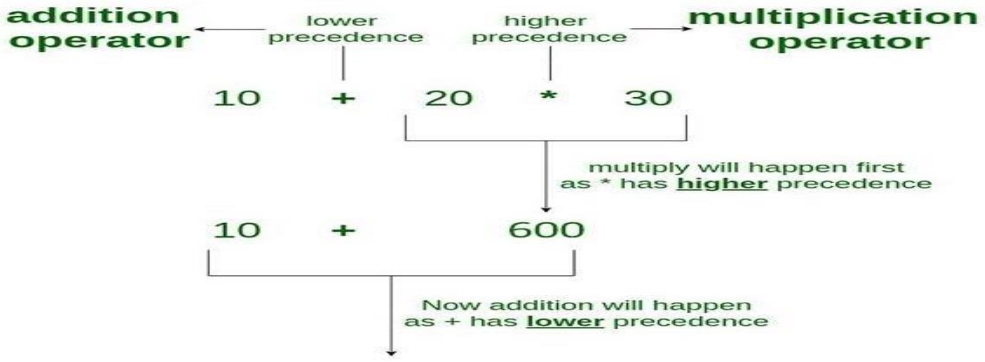
Describe in brief operator precedence and Associativity with examples?

Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence.

example:**10 + 20 * 30**

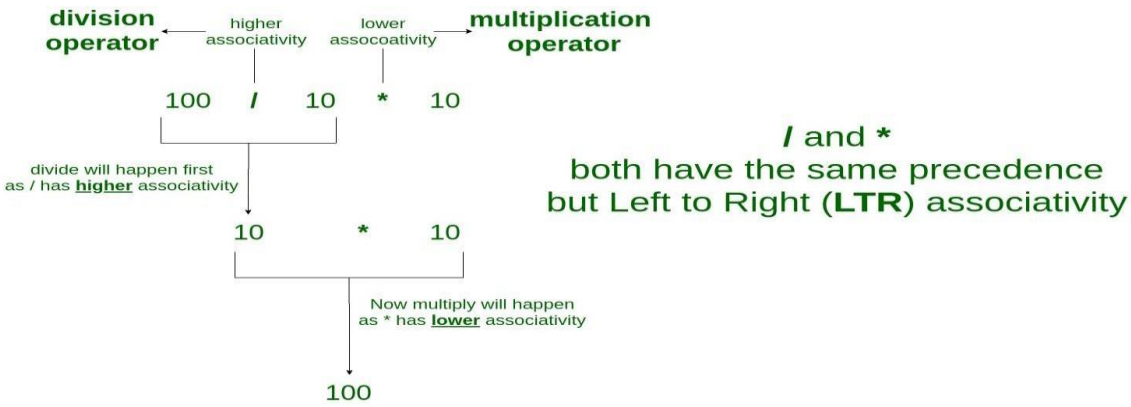
10 + 20 * 30 is calculated as **10 + (20 * 30) = 610**

and not as(10 + 20) * 30



Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either Left to Right or Right to Left.

For example: '*' and **'/'** have same precedence and their associativity is Left to Right, so the expression "100 / 10 * 10" is treated as "(100 / 10) * 10".



Operator	Description	Associativity
() [] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+-	Addition/subtraction	left-to-right

27.

Define String and Explain any 3 String handling functions used in C?

A string is **a sequence of characters terminated with a null character \0 .**

or

The string can be defined as the **one-dimensional array of characters terminated by a null ('\0').**

Syntax---

Char variable_name [array_size];

For example:

```
char c[] = "c string";
char name[30] = "LORDS";
```

i. strcpy()-- copying source string into destination string.

syntax-strcpy(Destination string, Source String);

ii. strcat() -- It combines two strings and store the result in Destination String.

syntax-strcat (Destination String, Source string);

iii. strrev()--The function is used for reversing a string. reversed string will be stored in the same string.

syntax-strrev (string)

iv. strlen()- It returns the number of characters in a string.

syntax- int strlen (string name);

28.

Write a C program to check whether given number is divisible by 5 and 7.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int num;
clrscr();
printf("\n enter any number");
scanf("%d", &num);
if((num%5 ==0)&&(num%7 ==0))
{
printf("\n given number is divisible by 5 and 7");
}
else
   {
printf("\n given number is not divisible by 5 and 7");
}
getch();
}
```

Output:

Enter any number: **35**

given number is divisible by 3 and 5

29.

Write a C program to check if the given string is palindrome or not.

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

int main()
{
   char str[100];
   int i, len, flag = 0;

printf("Enter a string: ");
   gets(str);

len = strlen(str);
for(i = 0; i<len; i++)
   {
      if(str[i] != str[len - i - 1])
      {
         flag = 1;
         break;
      }
   }
   if (flag)
printf("%s is not a palindrome.\n", str);
 else
printf("%s is a palindrome.\n", str);
   return 0;
}
```

Output:

```
 Enter a string: MADAM
MADAM is a palindrome
```

30.

Explain in brief about Switch case statement in C.

the switch statement is a control statement that allows you to choose between multiple execution paths based on the value of a variable or expression. The switch statement is typically used as an alternative to long if-else chains.

SYNTAX

```
switch (expression)
{
   case constant1:      // code to execute if expression == constant1
      break;
   case constant2:  // code to execute if expression == constant2
  break;
// more cases...
   default:       // code to execute if expression doesn't match any case
}
```

PROGRAM

```
#include <stdio.h>
int main()
{
   int num;
printf("Enter a number between 1 and 3: ");
scanf("%d", &num);
```

```
    switch (num)
    {
        case 1:
                printf("You entered 1.\n");
                break;
        case 2:
                printf("You entered 2.\n");
                break;
        case 3:
                printf("You entered 3.\n");
                break;
        default:
                printf("Invalid input.\n");
                break;
    }
    return 0;
}
```

UNIT-III

31. **Write Linear search algorithm and Explain how it works with example**

Linear search is a simple algorithm that searches for a specific value in a list or array by iterating through each element of the list until the desired value is found or the end of the list is reached.

Linear Search Algorithm

Array=A[n];**n=size of an array ;**

X= Search element to find its position

Step 1– Set **i=0,**

Step 2—Check ifi>n goto Step 5

Step 3—if A[i] == X

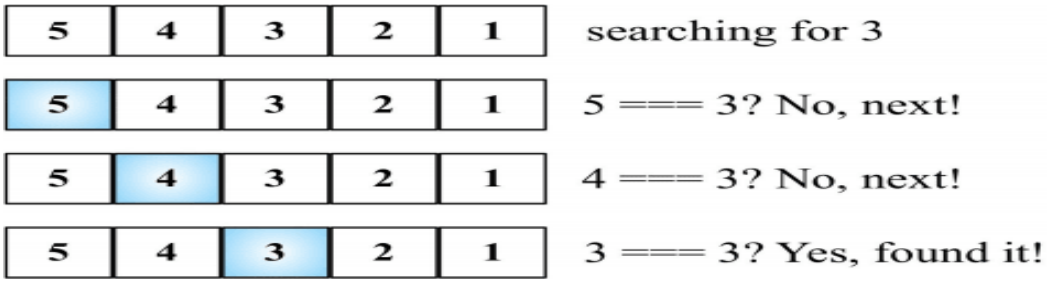
Pos= mid and goto Step 6

Step 4 – else

i=i+1 & goto Step 2

Step 5 -- Element not found in List and Exit.

Step 6 -- Print Element found at Pos and Exit.



32.

Write an algorithm to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).

ALGORITHM

Step 1. Start
Step 2. Read the coefficients of the equation, a, b and c from the user.
Step 3. Calculate **discriminant (d)**= (b * b) – (4 * a * c)
Step 4. **If d> 0**:
r1 = (-b + sqrt(**d**)) / (2 * a)
r2 = (-b - sqrt(**d**)) / (2 * a)
Display " Roots **r1** and **r2** are **real and different**"
Step 5: **Else ifd = 0:**
r1 = -b / (2 *a)
r2 = **r1**
Display "Roots**r1** and **r2**are **real and equal**"
Step 6.Else:
Calculate **real** = -b / (2 * a)
Calculate **imaginary** = sqrt(-discriminant) / (2 * a)
Display "**Roots are imaginary**"
Display real, **"±"** , imaginary, **"i"**

Step 7. Stop

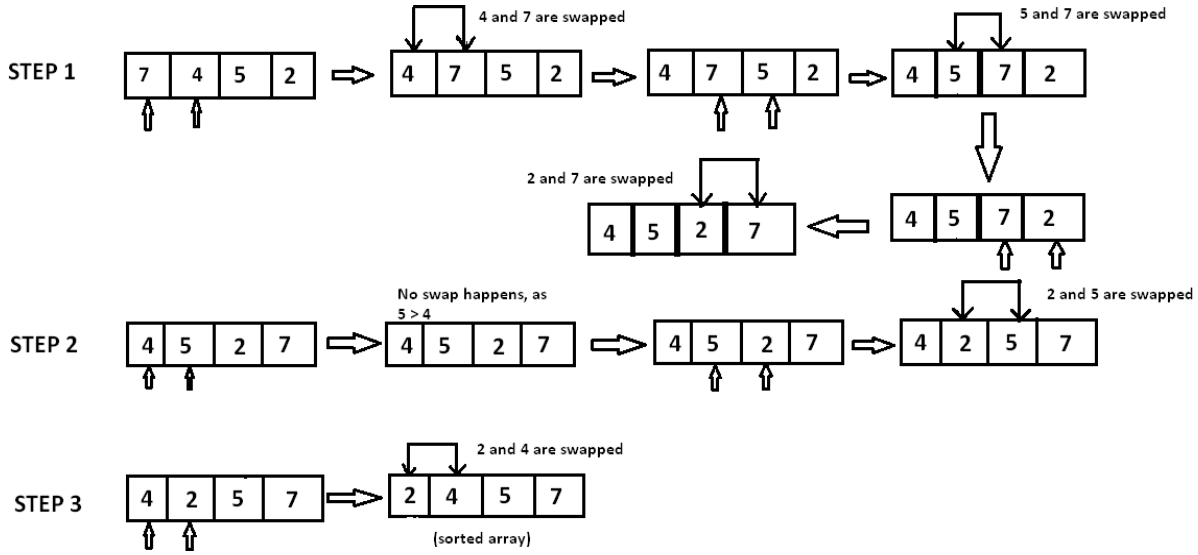
33.

Write Bubble sort algorithm and explain how bubble sort works with example.

BUBBLE SORT --- It is the simplest sorting algorithm that runs through the list repeatedly, compares adjacent elements, and swaps them if they are out of order.

Bubble Sort Algorithm

Step 1--Look at the first number in the **Array/List**.
Step 2--**Compare** the current number with the next number.
Step 3--Is the next number smaller than the current number? If so, **swap** the two numbers around. If not, do not **swap**.
Step 4--**Move to the next number along in the list and make this the current number**.
Step 5--Repeat from **step 2** until the last number in the list has been reached.
Step 6--If any numbers were **swapped**, repeat again from **step 1.**
Step 7--If the end of the list is reached without any **swaps** being made, then the **list is ordered and the algorithm can stop.**



34.

Write Selection sort algorithm and Explain how it works with example.

SELECTION SORT --- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning.

Selection Sort Algorithm

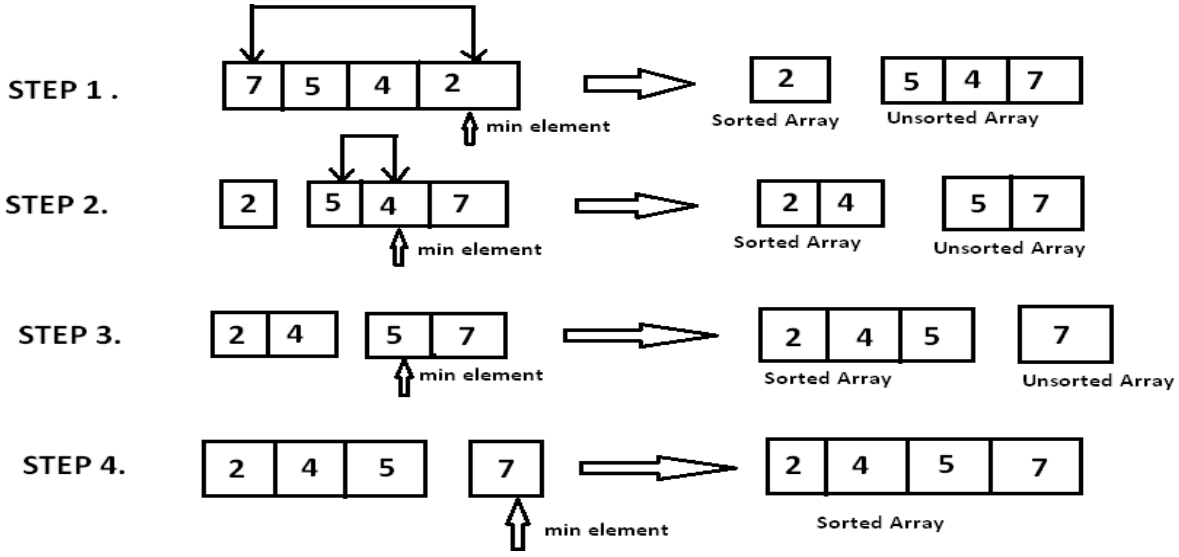
Step 1: Set the first element as **minimum** (i.e., Element at first position in the list).

Step 2: **Compare** **minimum** with the second element. If the second element is smaller than minimum, assign the second element as minimum.

Step 3: **Compare** **minimum** with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. **The process goes on until the last element.**
Step 4: After each iteration, **minimum** is placed in the front of the unsorted list.

Step 5: For each iteration, indexing starts from the first unsorted element. **Step 1** to **3** are repeated **until all the elements are placed at their correct positions**.

Example



35.

Explain how binary search works to search a number (25) in the given list

45 37 31 28 25 15 7 3.

ArrayA[n], n=7
Search Element= **Key**= **25**
Low=0
High=7

45	37	31	28	25	15	7	3	
Low=0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	= **High**

Step-1:

Mid=(Low + high)/2=>Mid=(0+7)/2=> 3.5 (discard fractional part)
Mid=3

45	37	31	28	25	15	7	3	
Low=0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	= **High**
			Mid					

if(A[Mid]==key)→if(A[3]== 30)→if(28==25)XnotFound
Check **Key<A[Mid]orkey>A[Mid]**

Step-2:

Key <A[Mid]

Then**Low =Mid+1**=>Low=3+1=>**Low=4**

45 37 31 28 25 15 7 3
0 1 2 3 4 5 6 7 = **High**
Low

Mid=(Low +high)/2
Mid=(4+7)/2=> 5.5 (discard fractional part)
Mid=5

45 37 31 28 25 15 7 3
0 1 2 3 4 5 6 7 = **High**
Low Mid

if(A[Mid]==key)→if(A[5]==25)→if(15==25)X not Found

Step-3:

Key >A[Mid]

Then**High =Mid-1**=> High=5-1=>**High=4**

45 37 31 28 25 15 7 3
0 1 2 3 4 5 6 7
Low
High

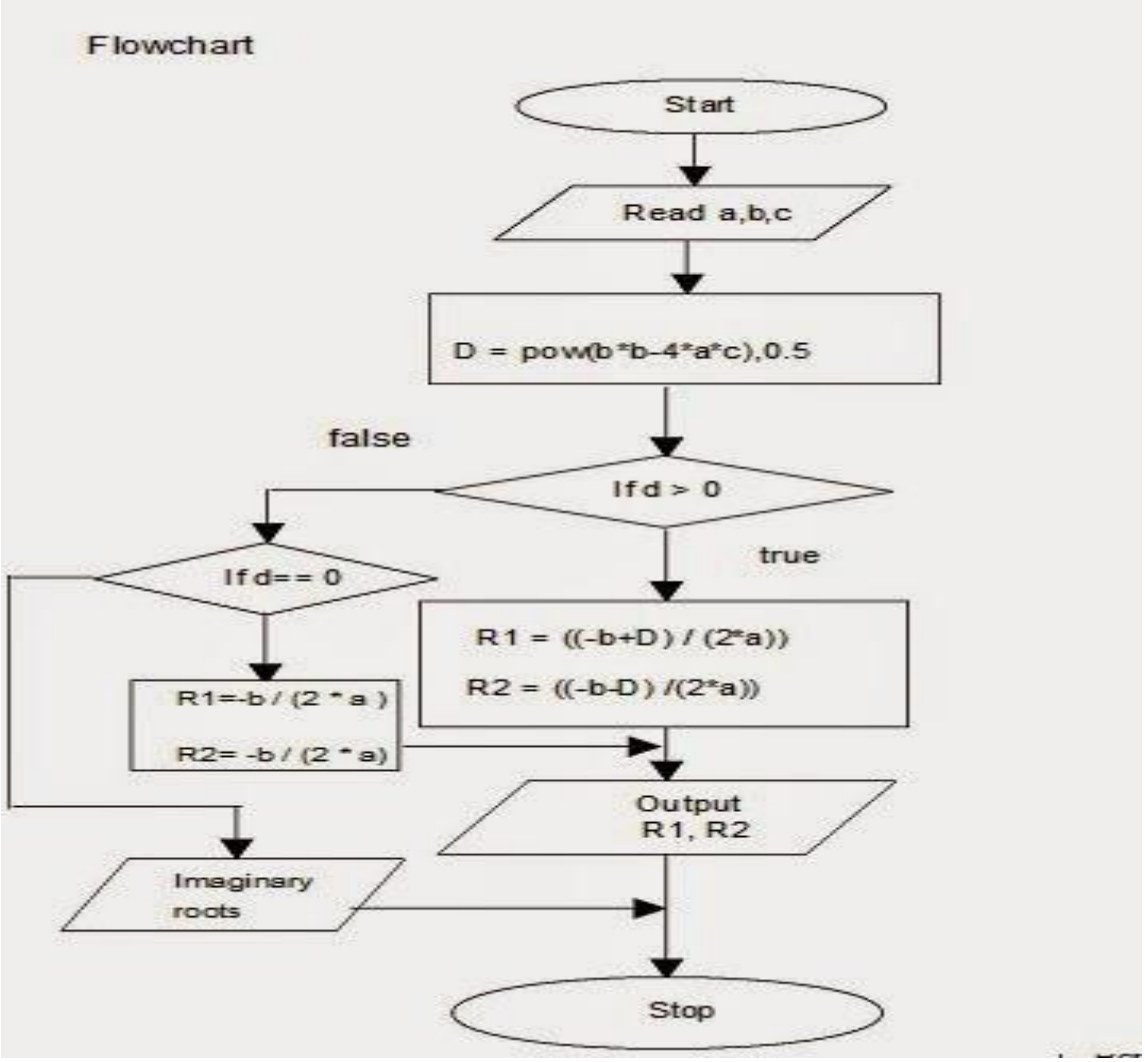
Mid=(Low + high)/2=>Mid=(4+4)/2=>Mid=8/2=>Mid=4

45 37 31 28 25 15 7 3
0 1 2 3 4 5 6 7
Low
High
Mid

if(A[Mid]==key)→if(A[4]==25)→if(25==25)

Search key is Found at position 4 of Mid (4) value.

36. **Draw a flowchart to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).**



37. **Write a C program to convert Binary to Decimal using functions.**

```
#include <stdio.h>
#include <math.h>
int convert(long long x); //function declaration
int main()
{
   long long x;
   printf("-----Enter a binary number to convert---- \n");
   scanf("%lld", &x);
   printf("\nThe Binary: %lld => Decimal: %d\n", x, convert(x));
   return 0;
}
```

```
int convert(long long x)  // function will convert number binary to decimal
{
    int dec = 0, i = 0, rem;
    while (x != 0)
     {
        rem = x % 10;
        x /= 10;
        dec += rem * pow(2, i);
        ++i;
    }
    return dec;
}
```

38. **Write a C program to find the roots of given Quadratic Equation ($ax^2 + bx + c = 0$).**

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

int main()
{

float a, b, c, dis, r1, r2;
printf("Enter coefficients a, b and c: ");
scanf("%f%f%f", &a, &b, &c);
dis = b*b - 4*a*c;

   // check if dis is positive, negative, or zero
   if (dis> 0)
{
r1 = (-b + sqrt(dis)) / (2*a);
r2 = (-b - sqrt(dis)) / (2*a);
printf("Roots are real and different.\n");
printf("r1 = %.2f and r2 = %.2f", r1, r2);
   }
   else if (dis == 0)
{
r1 = r2 = -b / (2*a);
printf("Roots are real and same.\n");
printf("r1 = r2 = %.2f", r1);
   }
   Else
{
      float realPart = -b / (2*a);
      float imagPart = sqrt(-dis) / (2*a);
printf("Roots are complex and different.\n");
printf("r1 = %.2f + %.2fi and r2 = %.2f - %.2fi", realPart, imagPart, realPart, imagPart);
   }
   return 0;
}
```

LORDS INSTITUTE OF ENGINEERING & TECHNOLOGY

Estd. 2002

UGC Autonomous

Approved by AICTE | Affiliated to Osmania University

Accredited by NBA | Accredited 'A' grade by NAAC |Certified by ISO.

Department of Science and Humanities

Academic Year: 2023-24

Year: I Semester: I

CIE II QUESTION BANK

PROGRAMMING FOR PROBLEM SOLVING

[U23CS101]

[Common to CSE,CSD,CIVIL,MECH]

Prepared by:

Course Coordinator:

Ms. Pooja Chavan,
Assistant Professor,
CSE Department.

Course Faculties:

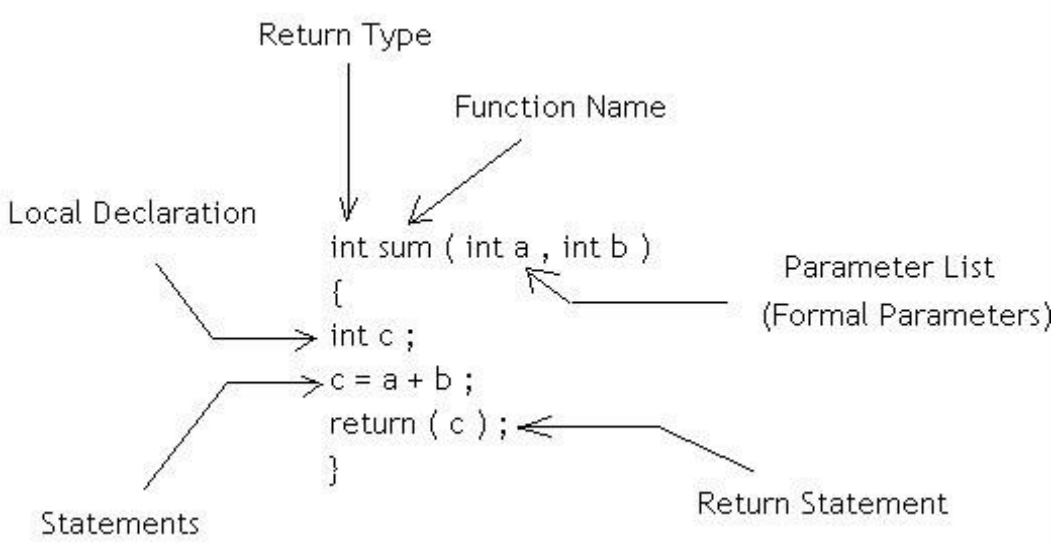
Mr. Amer Noor Khan, Assistant Professor
Mrs. K Shilpa, Assistant Professor
Mrs. M. Kavitha, Assistant Professor
Mrs. D. Latha, Assistant Professor

Note: A question bank is versatile and flexible FAQs that cover the entire syllabus of a subject. It is used by students and teachers for learning and assessment purposes only.

SAQs

1. **Define a function definition with an example.**

A function is a self-contained block of code that performs a specific task.These functions are known as user-defined functions.
Function definition is a actual code of instructions/ statements written in a program.



2. **Define function? List out the advantages of functions in C.**

In c, we can divide a large program into the basic building blocks known as function.
A function is a self-contained block of code that performs a specific task.
Advantage of functions in C

- By using functions, we can avoid repetition of codes.
- Increases program readability.
- Reusability.
- Reduces chances of error.
- Modifying a program becomes easier by using function.

3. **What are actual and formal parameters in function calls?**

Parameter acts as variables inside the function.
Parameters are specified after the function name, inside the parentheses**()** .
Actual Parameters-- Actual parameters are values that are passed to a function when it is invoked. These can be variables, constants, and expressions, without data types.
Formal Parameters--Formal parameters are values that are received by a function when it is invoked. Formal parameters are variables with the data type.

4. **Write a C Function to calculate average of three numbers.**

```
float average(float num1, float num2, float num3)
{
  return (num1 + num2 + num3) / 3.0;
}
```

This function takes three input numbers as arguments (num1, num2, and num3), calculates their sum, and then divides by 3 to get the average. The float type is used to represent the input and output values as decimal numbers.

5. **Compare between Call by Value and Call by Reference.**

PARAMETERS	CALL BY VALUE	CALL BY REFERENCE
Basic	A copy of the variable is passed.	A variable itself is passed.
effect	Change in a copy of variable doesn't modify the original value of variable.	Change in a copy of variable modify the original value of variable.
Syntax	function_name(variable_name1, variable_name2...)	function_name(&variable_name1, &variable_name2...)
Default calling	Primitive type are passed using "call_by_value".	Objects are implicitly passed using "call_by_reference".

6. **Enlist various categories of User Defined Functions?**

All the **C** functions can be called either with **arguments/parameters** or without arguments/parameters in a C program. These functions may or may not **return values** to the calling function.

1. Function **without arguments** and **without** a **return** value.
2. Function **with arguments** and **without** a **return** value.
3. Function **without arguments** and **with** a **return** value.
4. Function **with arguments** and **with** a **return** value.

7. **What is meant by function prototype in C.**

Function Prototype/Declaration

A function prototype is simply the **declaration of a function that specifies function's name, parameters and return type.** It **doesn't contain function body**.

A function prototype gives information to the compiler that the function may later be used in the program.

Syntax-

```
return_type fun_name (list of Parameters);
```

Example-

```
int Factorial(int Number);
void product();
```

Unit-IV

8. **Write a short notes on enum data type?**

An enumeration is a data type that consists of a set of named values that represent integral constants, known as enumeration constants.

Syntax—

enum enum_name{ int_const1, int_const2, int_const3, …. int_constN };

Example--

```
enum days {S, M, T, W, Th, F, Sa};
void main()
{
int i; // printing the values of weekdays
for(i=S; i<=Sa; i++)
{
printf("%d, ",i);
}
}
```

Output: 0,1,2,3,4,5,6

9. **What are the advantages and disadvantages of recursion?**

S.No.	Advantages	Disadvantages
1.	Recursion makes code easier to write.	Recursive functions are generally slower than non-recursive function.
2.	Recursion reduces unnecessary calling of function.	Hard to analyze or understand the code.
3.	Recursion is very useful in solving recursive problems.	Need to identify the base case and general case.
4.	Used in stack data structure applications like infix to postfix conversion etc.	It is not efficient in terms of time and space complexity.

10. **What is the use of typedef in C language?**

Def--- typedef is a predefined keyword, which can replace the name of the existing data type with the new name.

Syntax --

```
typedef existing_name alias_name
```

Example--

```
#include <stdio.h>
typedef    int NUMBER;
typedef    float   FRACTION;
void main()
{
NUMBER      a = 5;
FRACTION      pi = 3.142;
printf("a and pi value is %d \t %d", a, pi);
}
```

11. **Write a recursive function to find Factorial of number.**

```
#include<stdio.h>
intfactorial( inti)
{
if(i<= 1)
        return1;
else
        returni*factorial(i - 1);
}
void main()
{
    int N =5;
printf("Factorial of %d is %d\n", N,factorial(N));
}
```

OUTPUT

Factorial of 5 is 120

12. **Differentiate between arrays and structures?**

S.No	Arrays	Structures
1	Array refers to a collection of similar data elements (homogeneous data type).	Structure refers to a collection of different data elements (heterogeneous data type).
2	Array uses subscripts or "**[ ]**" (square bracket) for element access	Structure uses "**.**" (Dot operator) for element access.
3	Arrays is a derived datatype.	Structures are user defined data type.
4	**Syntax--** **data_type array_name[size];**	**Syntax--** **struct structure_name** **{** **data_type1var1;** **data_type2var2;** **};**

13. **What are user defined data types in C?**

In C, user-defined data types are types that are defined by the user instead of being built-in to the language.

1. **structure:-** A structure is a user-defined data type that can hold multiple related variables of different data types in unique memory location .
2. **union:-** A union is a user-defined data type that can hold multiple related variables of different data types in same memory location .
3. **typedef:-** typedef is a predefined keyword, which can replace the name of the existing data type with the new name.
4. **enum:-** An enumeration is a user data type that consists of a set of named values that represent integral constants, known as enumeration constants.

14. **How to declare and access the members of structure?**

A structure is a user-defined data type that can hold multiple related variables of different data types.

Syntax--Example --

```
struct structure_name
{
  data_type1var1;
  data_type2var2;
}VAR1, VAR2;
```

```
struct Person
{
   char name[50];
   int age;
} p1,p2;  //DECLARING STRUCTURE VARIABLES
```

ACCESSING STRUCTURE MEMBERS

Using the dot operator:

```
struct Person
{
   char name[50];
   int age;
   float salary;
};
struct Person p1;
p1.age = 30;
printf("Age: %d", p1.age);
```

15. **Write the recursive function to find LCM of two numbers.**

int lcm_recursive(int a, int b, int i);

```
int lcm(int a, int b)
{
   return lcm_recursive(a, b, 1);
}
int lcm_recursive(int a, int b, int i)
{
   if (i % a == 0 &&i % b == 0)
         return i;
   else
    {
      return lcm_recursive(a, b, i+1);
    }
}
```

```
int main()
{
   printf("The LCM of 4 and 5 is %d\n", lcm(4,5));
   return 0;
}
```

16. **Differentiate between recursion and iterative statements.**

Iteration	Recursion
Iteration explicitly user a repetition structure.	Recursion achieves repetition through repeated function calls.
Iteration terminates when the loop continuation.	Recursion terminates when a base case is recognized.
Iteration keeps modifying the counter until the loop continuation condition fails.	Recursion keeps producing simple versions of the original problem until the base case is reached.
Iteration normally occurs within a loop so the extra memory assigned is omitted.	Recursion causes another copy of the function and hence a considerable memory space's occupied.
It reduces the processor's operating time.	It increases the processor's operating time.

17. **Write the syntax and example of Union variable.**

Union is an user defined datatype in C programming language. It is a collection of variables of different datatypes in the same memory location.

Syntax---Example---

```
union union_name
{
 data_type member1;
 data_type member2; ;
  ....     ;
  data_type member N;
};
```

```
union Time
{
  int Hour;
   int Minutes;
    float Seconds;
 }t1     = { 15, 45, 56.089};
  //t1 is union variable
```

18. **Write short notes on typedef data type?**

Def--- typedef is a predefined keyword, which can replace the name of the existing data type with the new name.

Syntax --

```
typedef  existing_name  alias_name
```

Example--

```
#include <stdio.h>
typedef    int       NUMBER;
typedef   float      FRACTION;
void main()
{
NUMBER  a=5;
FRACTION   pi=3.142;
}
```

19. **Define structure and create Employee structure (empid, esal, doj, ename).**

A structure is a user-defined data type that can hold multiple related variables of different data types

```
struct Employee
{
      int empid;
      float esal;
      char doj[10];
      char  ename[50];
} e1={ 555, 67895.890,"05/11/2002", "ISMAIL"};
```

20. **List any 3 differences between structure and union.**

S.NO	STRUCTURE	UNION
1	We use the struct statement to define a structure.	We use the union statement to define a union.
2	Every member is assigned a unique memory location.	All the data members share a memory location.
3	A structure's total memory size is the sum of the size of every data member.	A union's total memory size is the size of the largest data member.
4	Users can access or retrieve any member at a time.	User can access or retrieve only one member at a time.

21. **Write the syntax of nested structure.**

A nested structure in C is a **structure within structure**.
One structure can be declared inside another structure in the same way structure members are declared inside a structure.

Syntax-

```
structname_1                    structname_2
{                               {
     member1;                        member1;
     member2;                        member2;
      ...                             .....
                                  struct name_1var;   ←Structure within Structure
      member n;                       member n;
 } ;                             } ;
```

22. **Write an example for enumerated data type?**

An enumeration is a data type that consists of a set of named values that represent integral constants, known as enumeration constants.

Example--

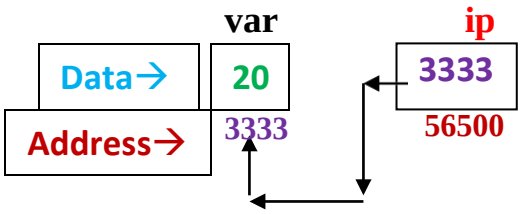
```
enum days {S, M, T, W, Th, F, Sa};
void main()
{
int i; // printing the values of weekdays
for(i=S; i<=Sa; i++)
{
printf("%d, ",i);
}
}
```

Output: 0,1,2,3,4,5,6

Unit-V

23. **How to initialize a pointer variable.**

Definition-- A pointer is a variable which stores the address location of another variable in memory.
Syntax---
Datatype *Variable;



Example—

```
#include <stdio.h>
void main ()
{
int var = 20;
int *ip;          /* Declaring pointer to an integer */
ip = &var;        /* Initializing a pointer variable */
printf("Address stored in ip variable: %u\n",ip);
}
```

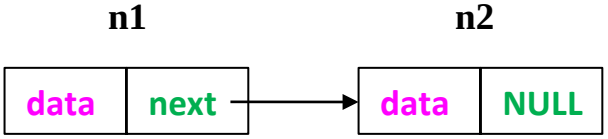
24. **Write advantages and disadvantages of pointers.**

S.No	Advantages	Disadvantages
1	Pointers provide *direct access to memory.*	Uninitialized pointers might cause *segmentation fault.*
2	Pointers allows us to perform dynamic memory allocation and deallocation.	*Memory leak* occurs if dynamically allocated memory *is not freed.*
3	Pointers helps us to build data structures like *linked list, stack, queues, trees, graphs* etc.	If pointers are updated with *incorrect values,* it might lead *to memory corruption.*
4.	They *increase the execution speed of the program.*	If sufficient memory is not available during runtime for the storage of pointers, the *program may crash*

25. **What is self referential structure?**

A structure can have data members which point to a structure variable of the same type. These types of structures are called self referential structures and are widely used in dynamic data structures like trees, linked list, etc.

```
struct node
{
    int data;
   struct node *next;
}n1,n2;
```



Here, **next** is a pointer to a **struct node** variable.

26. **What is a FILE and explain types of Files used in C?**

File is a collection of data that stored on secondary memory like hard disk of a computer
C programming language supports two types of files.
1--*Text Files (or) ASCII Files*
2--*Binary Files*

Text File (or) ASCII File - The file consists ASCII code of data like digits, alphabets and symbols is called text file (or) ASCII file. **Eg--** e**.txt, .c, .h, .java, .py**etc.,

Binary File - The file contains data in the form of bytes (0's and 1's) is called as binary file. Generally, the binary files are compiled version of text files. **Eg-.bin, .bat** etc.

27. ## Write the various text file opening modes?

We use the pre-defined method **fopen()** to open an existing file. which has different modes.

Mode	Description
r	opens a text file in reading mode
w	opens or create a text file in writing mode.
a	opens a text file in append mode
r+	opens a text file in both reading and writing mode
w+	opens a text file in both reading and writing mode
a+	opens a text file in both reading and writing mode
rb	opens a binary file in reading mode
wb	opens or create a binary file in writing mode
ab	opens a binary file in append mode
rb+	opens a binary file in both reading and writing mode
wb+	opens a binary file in both reading and writing mode
ab+	opens a binary file in both reading and writing mode

28. ## Write syntax and example of fprintf ().

fprintf()

fprintf()is a standard C library **function that is used to write set of characters into a file.**

Syntax---

int **fprintf**(FILE ***stream**, const char ***format**, **...**)

stream--- this is the pointer to a FILE object that identifies the stream.
format --- This is the string that contains the text to be written to the stream.

EXAMPLE-----

```
#include <stdio.h>
void main()
{
  FILE *fptr;
fptr = fopen("output.txt", "w");
  char str[] = "Hello, world!";
fprintf(fptr, "str = %s\n", str);
fclose(fptr);
}
```

29. **Write the syntax and usage of rewind ()?**

rewind()

The rewind() **function sets the file pointer at the beginning of the stream**. It is useful if you have to use stream many times.

Syntax---

void **rewind**(FILE ***stream**)

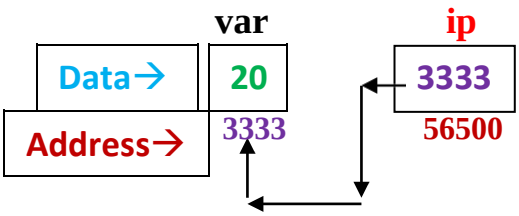
stream--- The stream whose file position indicator is to be set to the beginning of the file.

30. **Define pointer and explain how to declare an integer pointer.**

Definition-- A pointer is a variable which stores the address location of another variable in memory.

Syntax---

Datatype *Variable;



Example—

```
#include <stdio.h>
void main ()
{
int var = 20;
int *ip;    /* Declaring pointer to an integer */
ip = &var;/* Initializing a pointer variable */
printf("Address stored in ip variable: %u\n",ip);
}
```

31. **Write the applications of Linked Lists?**

APPLICATIONS OF LINKED LIST

- Implementation of **stacks** and **queues**
- Implementation of **graphs**
- **Dynamic** memory allocation and deallocation.
- Manipulation of polynomials
- ***Image viewer***
- ***Previous and next page in web browser***
- ***Music Player with previous and next Songs playing list***

32. **Is null pointer same as uninitialized pointer?**

No, a null pointer and an uninitialized pointer are not the same.

A **null pointer** is a special value that points to no memory location. It is often used to indicate that a pointer does not currently point to a valid object.

In C ,a null pointer is typically represented by the value 0 or the macro NULL.

An uninitialized pointer is a pointer that has not been assigned a value. It may contain a garbage value, which could be any memory address.

Using an uninitialized pointer can lead to undefined behaviour, as the pointer may point to an unexpected location in memory.

33. **Explain different types of Files used in C?**

File is a collection of data that stored on secondary memory like hard disk of a computer

C programming language supports two types of files.

1--*Text Files (or) ASCII Files*

2--*Binary Files*

Text File (or) ASCII File - The file consists ASCII code of data like digits, alphabets and symbols is called text file (or) ASCII file. **Eg--** e**.txt, .c, .h, .java, .py**etc.,

Binary File - The file contains data in the form of bytes (0's and 1's) is called as binary file. Generally, the binary files are compiled version of text files. **Eg-.bin, .bat** etc.

34. ## Write the syntax and example to open a file.

In C programming, the fopen() function is used to open a file for reading or writing. It returns a FILE* pointer that can be used to read from or write to the file.

syntax:-

```
FILE *fopen(const char *filename, const char *mode);
```

Example:-

```
#include <stdio.h>
voidmain()
{
   FILE *fp;
  fp = fopen("input.txt", "r");
if(fp == NULL)
{
printf("Error opening file\n");
      return 1;
}
fclose(fp);
}
```

35. ## List the applications of Pointers.

Pointers are a powerful feature of the C programming language which has following applications

1. Dynamic memory allocation
2. Passing arguments to functions
3. Returning multiple values from a function
4. Implementing data structures
5. Low-level memory manipulation

36. ## Write syntax and example of fscanf ().

The fscanf() function in C is used to read formatted input from a file. It reads input from a specified stream (FILE*) and stores the formatted values into the specified variables.

Syntax:-

```
int fscanf(FILE* stream, const char* format, ...);
```

Example

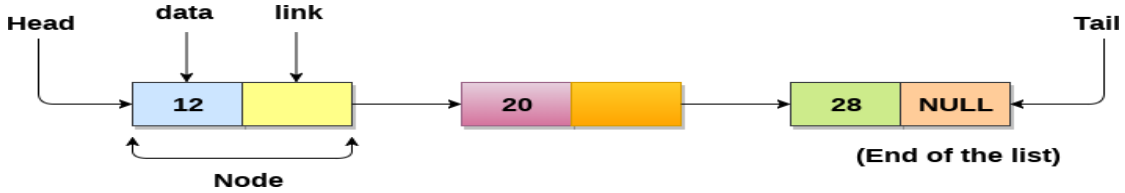
```
#include <stdio.h>
voidmain()
{
   FILE* fp;
   int num1, num2;

fp = fopen("input.txt", "r");
fscanf(fp, "%d %d", &num1, &num2);    // Read two integers from the file

printf("Read values: %d %d\n", num1, num2);// Print the values read from the file
fclose(fp);
}
```

37. **Define linked list with an example.**

A linked list is a linear data structure in which elements, called nodes, are stored in a non-contiguous manner. Each node contains two parts: data and a pointer to the next node in the list. The first node is called the head, and the last node points to null.



LAQs

UNIT III

39. **Write a C program to Swap two numbers using call by reference.**

```
#include<stdio.h>
#include<conio.h>
void swap(int *x, int *y);
void main()
{
        int a,b;
        clrscr();
        printf("Enter a and b values\n");
        scanf("%d%d",&a,&b);
        printf("Before swap a=%d and b=%d\n",a,b);
        swap(&a , &b);
        printf("After swap a=%d and b=%d\n",a,b);
        getch();
}
void swap(int *x, int *y)
{
int temp;
temp = *x;
*x=*y;
*y=temp;
}
```

Output:

```
Enter a and b values
20      30
Before swap, a=20 and b=30
After swap, a=30 and b=20
```

40.

Explain different ways of passing arguments to function with an example.

Arguments is also referred as parameters in C. There are two Arguments passing techniques in C.

1) Call/Pass by Value

2) Call/Pass by Reference

While calling a function, we pass values of variables to it. Such functions are known as **"Call By Values"**

While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as **"Call By References"**

//CALL BY VALUE Program

```
#include<stdio.h>
void swap(int a, int b)
{
Int temp;
 temp=a;
a=b;
b=temp;
}
void main()
{
 int a=100, b=200;
printf("\n Before Swap Value a=%d, b= %d", a, b);
swap(a, b);// passing value to function
printf("\n After Swap Value a=%d, b= %d", a, b);
}
```

Output:

Before Swap Value a=100, b=200

After Swap Value a=100, b= 200

//CALL BY REFERENCE Program

```
#include<stdio.h>
 void swap(int *a, int *b)
{
    int temp;
     temp=*a;
    *a=*b;
    *b=temp;
}
 void main()
 {
     int a=100, b=200;
     printf("\n Before Swap Value a=%d, b= %d", a, b);
    swap(&a, &b); // passing addresses to function
    printf("\nAfter Swap Value a=%d, b= %d",a,b);
}
```

Output:

Before Swap Value a=100, b= 200

After Swap Value a=200, b= 100

41.

Illustrate various categories of user defined functions in brief.

All the **C** functions can be called either with **arguments/parameters** or without arguments/parameters in a C program. These functions may or may not **return values** to the calling function.

1. Function **without arguments** and **without** a **return** value.
2. Function **with arguments** and **without** a **return** value.
3. Function **without arguments** and **with** a **return** value.
4. Function **with arguments** and **with** a **return** value.

```
#include<stdio.h>
void Sum();                 //Function without arguments and without a return value
void Square(int );          //Function with arguments and without a return value
int Product();              //Function without arguments and with a return value
int  Big(int , int);        //Function with arguments and with a return value
```

```
void main()
{
        int a=5, b=7, result1, result2;
        Sum();
        Square(a);
        result1 = Product();
        result2 = Big(a, b);
         printf(" \n Biggest number is %d", result2);
         printf(" \n Product is %d", result1);
}
```

```
void Sum()
 {
    int x=90, y=60;
     printf(" \n Sum is %d", x+y);
 }
```

```
 int Big( int p, int q)
{
      int res = (p>q)? p : q;
     return res;
}
```

```
void Square(int m)
{
    int res= m*m;
     printf(" \n Square is %d", res);
}
```

```
int Product( )
{
     int m=6,n= 8;
     int res = m*n;
     return res;
}
```

OUTPUT:

Sum is 150

Square is 25

Biggest Number is 7

Product is 48

42.

Describe about call by value and call by reference in C.

There are two Parameter passing techniques in C. they are

1) Call/Pass by Value

2) Call/Pass by Reference

While calling a function, we pass values of variables to it.

Such functions are known as **"Call By Values"**

While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as **"Call By References"**

//CALL BY VALUE Program

```
#include<stdio.h>
void swap(int a, int b)
{
  Int temp;
  temp=a;
  a=b;
  b=temp;
}
void main()
{
 int a=100, b=200;
printf("\n Before Swap Value a=%d, b= %d", a, b);
swap(a, b);// passing value to function
printf("\n After Swap Value a=%d, b= %d", a, b);
}
```

Output:

Before Swap Value a=100, b=200

After Swap Value a=100, b= 200

//CALL BY REFERENCE Program

```
#include<stdio.h>
 void swap(int *a, int *b)
{
        int temp;
       temp=*a;
       *a=*b;
       *b=temp;
 }
 void main()
 {
     int a=100, b=200;
     printf("\n Before Swap Value a=%d, b= %d", a, b);
     swap(&a, &b);// passing addresses to function
     printf("\nAfter Swap Value a=%d, b= %d",a,b);
 }
```

Output:

Before Swap Value a=100, b= 200

After Swap Value a=200, b= 100

43.

Write a C program to convert Decimal to Binary using functions.

```
#include <stdio.h>
void decToBinary(int n); // function prototype
void  main()
{
   int decimal;
printf("Enter a decimal number: ");
scanf("%d", &decimal);
printf("Binary equivalent: ");
decToBinary(decimal); //Function call
}
void decToBinary(int n) // function definition  to convert decimal to binary
{
   int binary[32], i = 0;
while (n > 0)
{
      binary[i] = n % 2;
       n /= 2;
       i++;
 }
   // print binary number in reverse order
   for (int j = i - 1; j >= 0; j--)
        printf("%d", binary[j]);

}
```

44.

Explain function call, function definition and function prototype in C program.

Definition--A function is a self-contained block of code that performs a specific task.These functions are known as user-defined functions.

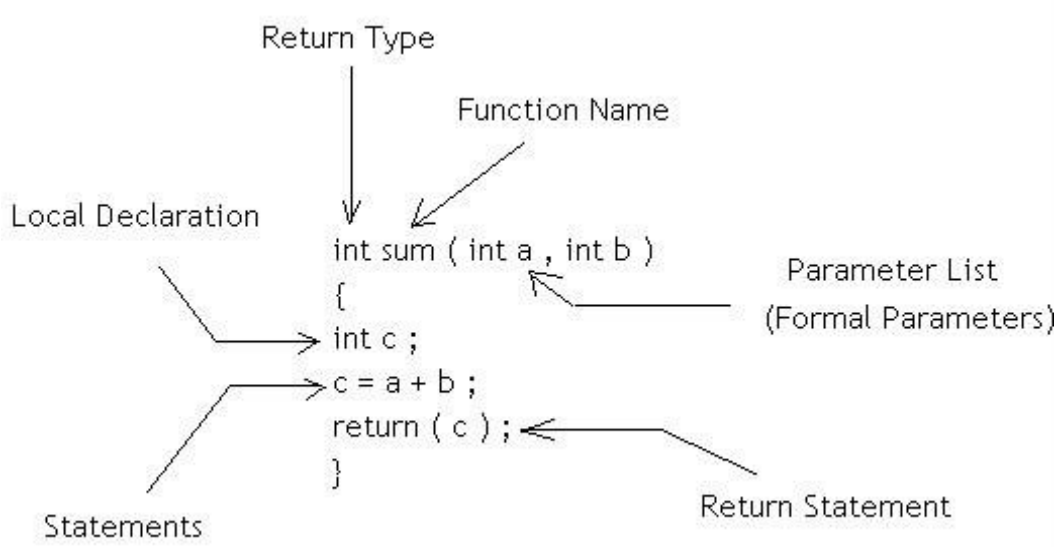
Function Call----

A function call is an expression containing the function name followed by the function call operator**parenthesis, ()** .

Eg-- main(), clrscr(), printf(),scanf(), etc.

Function Definition----

Function definition is a actual code of instructions/ statements written in a program.



Function Prototype/Declaration

A function prototype is simply the **declaration of a function that specifies function's name, parameters and return type.** It **doesn't contain function body**.

A function prototype gives information to the compiler that the function may later be used in the program.

Syntax-

return_typefun_name(list of Parameters);

Example-

intFactorial(int Number);

voidproduct();

45. **Explain how arrays are passed to a function with a program.**

```
// Program on Sum of all Elements in an Array.
#include<stdio.h>
voidsum(int [ ]);
void main( )
{
int marks[5], i;
marks[5] ={10, 20, 30, 40, 50};
sum(marks);
}
voidsum(intn[ ])
{
int i, sum=0;
for(i=0; i<5; i++)
{
 sum = sum+n[i];
}
printf("Sum = %d ", sum);
}
```

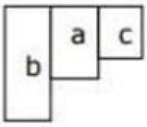
OUTPUT

Sum = 150

UNIT-IV

46. **Differentiate between structure and union with suitable examples.**

Structure	Union
Each member is assigned its own unique storage area.	All members share the same storage area.
Total memory required by all members is allocated.	Maximum memory required by the member is allocated.
All members are active at a time.	Only one member is active a time.
All members can be initialized.	Only the first member can be initialized.
Requires more memory.	Requires less memory.

Example:
struct SS
{
int a;
float b;
char c;
};
1 byte for c
2 bytes for a
4 bytes for b
	Example:
	union UU
	{
	int a;
	float b;
	char c;
	};
	4 bytes for c,b,a
	 

Total bytes = 1 + 2 + 4 = 7 bytes.	4 bytes are there between a,b and c because largest memory occupies by float which is 4 bytes.

47.

Using recursion write a C program to display the sum of digits of a given number.

```
#include <stdio.h>
int sum (int a);
void main()
{
int num, result;
printf("Enter the number: ");
scanf("%d", &num);
result = sum(num);
printf("Sum of digits in %d is %d\n", num, result);
}

int  sum (int num)
{
    if (num!= 0)
        return (num% 10 + sum (num/ 10));
    else
        return 0;
 }
```

Output--

Enter the number:234

Sum of digits in234 **is** 9

48.

What is a Union? Explain it with a program.

Union is an user defined datatype in C programming language. It is a collection of variables of different datatypes in the same memory location.

Syntax---

```
union union_name
{
 data_type member1;
data_type member2;
....
data_type memeberN;
};
```

C Program

```
union employee
{
  int id;
char name[50];
 } e1 ;
void main( )
{
e1.id=101;
strcpy( e1.name , "MUTEEB");
union employee e2;
printf( "employee 1 id : %d\n", e1.id );
printf( "employee 1 name : %s\n", e1.name );
}
```

All the data members share the same storage area. Memory size allocation is depends on **Largest memory size**of data member will be allocated in **union** Data type.

e1 is a variable of**employee union data type** which has **memory size of 50 bytes which is for all data members is allocated where as if we create Date data type using Structure will have 52 bytes .**

49.

Write a C program to display the Fibonacci series till Nth term using recursion.

```
#include<stdio.h>
int fib(int n)
{
if(n == 0)
    {
return 0;
    }
if(n == 1)
   {
return 1;
   }
return fib(n-1) + fib(n-2);
}
void main()
{
int N =7 ,i;
printf("Fibonacci series  \n");
   for (i=0; i<N; i++)
   {
printf(" %d \t",fib(N));
}
```

OUTPUT

```
Fibonacci series
0   1    1    2    3    5    8
```

50.

Write a C program to store 5 fields (Name, Roll Number, Branch, Subject and Marks)of students and display the same using structure.

```
#include <stdio.h>
#include<string.h>
struct Student
{
        long int rollnum;
        char name[30];
        char Branch[20];
        char Subject[200];
        float Marks[5];
};
void main()
{
        int i;
        char ch;
        structStudents1;

strcpy(s1.name," ISMAIL ");
s1.rollnum= 564;
strcpy(s1.Branch," CSE ");
strcpy(s1.Subject," C Programming ");
        s1.Marks[0]=90.2;        s1.Marks[1]=88;        s1.Marks[2]=77.29;
        //Displaying Students details
```

```
        printf("\n---------- Displaying Students Details ---------- \n");
        printf("Name \t: %s \n",s1.name);
        printf("roll number \t: %ld \n",s1.rollnum);
        printf("Branch \t: %s \n",s1.Branch);
printf("Subject \t: %s \n",s1.Subject);
printf(" *******Marks level wise ****** \n: %s \n",s1.Address);
printf("Level 1 \t: %f\n Level 2 \t: %f\nLevel 3 \t: %f\n",s1.Marks[0],s1.Marks[1], s1.Marks[2]);
}
```

Output:

```
---------- Displaying Students Details ----------
Name:ISMAIL
roll number:564
Branch:CSE
Subject         :C Programming
*******Marks level wise ******
Level 1:90.20000
Level 2:88.00000
Level 3:77.29000
```

51.

Explain about Recursive functions with example.

In C Programming, if a function calls itself from inside the same function is called **recursion**.

Recursive Function

The **function which calls itself is called a recursive function** and the function call is termed a recursive call. And Recursions in C is alternative to the Loops concept.

Syntax---

```
Return_typerecursion(List of Parameters)
{
recursion();  /* function calls itself */
}
void main()
{
recursion();
}
```

example--

```
#include <stdio.h>
void main()
{
printf("Hi it is RECURSION\n");
main();
 }
```

```
OUTPUT
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
Hi it is RECURSION
.
.
Infinite times it prints NO Stop
```

52.

Illustrate different ways to initialize and access the members of a structure?

In C, a structure is a user-defined data type that can hold multiple related variables of different data types. Here are some ways to initialize and access the members of a structure:

ACCESSING STRUCTURE MEMBERS

1. Using the dot operator:

```
struct Person
{
   char name[50];
   int age;
   float salary;
};
```

```
struct Person p1;
p1.age = 30;
printf("Age: %d", p1.age);
```

2. Using a pointer to a structure with the -> operator:

```
struct Person
{
  char name[50];
  int age;
  float salary;
};
struct Person p1 = {"John", 30, 5000.50};
struct Person *ptr = &p1;
printf("Name: %s", ptr->name);
```

INITIALIZATION OF A STRUCTURE

1. Initializing structure members individually:

```
struct Person
{
  char name[50];
  int age;
  float salary;
};
struct Person p1;
strcpy(p1.name, "John");
p1.age = 30;
p1.salary = 5000.50;
```

2. Initializing structure members while declaring the structure:

```
struct Person
{
  char name[50];
  int age;
  float salary;
} p1 = {"John", 30, 5000.50};
```

53.

Explain how Array of Structure is used in a Program

Array of Structure

An array of structures in C can be defined as the collection of multiple structures variables where each variable contains information about different entities.
The array of structures in C are used to store information about multiple entities of different data types.

```
structStudent
{
int rollno;
char name[10];
};
   void main()
   {
        struct StudentS[5];
       S[0].rollno=777;        S[1].rollno=999;S[2].rollno=555;
      strcpy(S[0].name, "Ravi");     strcpy(S[1].name, "Adil");strcpy(S[2].name, "Khan");
   }
```

← Array of Structure declaration

In above program **Student** is created as User defined data type and **S[5]** is an Array of Structure Variable which can hold maximum 5 Structure variable data is called as **Array of Structure.**

54.

Write a C program to display the GCD of two numbers using recursion.

```
#include<stdio.h>
#include<conio.h>
int gcd(int n1,int n2);
void main()
{
        int a, b, result;
        clrscr();
        printf("enter a and b values\n");
        scanf("%d%d",&a,&b);
        result=gcd(a,b);
        printf("GCD of %d and %d is %d\n",a,b,result);
        getch();
}
int gcd(int n1,int n2)
{
        if(n2!=0)
            return gcd(n2,n1%n2);
        return n1;
}
```

Output:

```
enter a and b values
40      20
GCD of 40 and 20 is 20
```

55.

How is the structure data type different from an array? Justify with a program.

S.No	Arrays	Structures
1	Array refers to a collection of similar data elements (homogeneous data type).	Structure refers to a collection of different data elements (heterogeneous data type).
2	Array uses subscripts or "**[ ]**" (square bracket) for element access	Structure uses "**.**" (Dot operator) for element access.
3	Arrays is a derived datatype.	Structures are user defined data type.
4	**Syntax--** **data_typearray_name[size];**	**Syntax--** **struct structure_name** **{** **data_type1var1;** **data_type2var2;** **};**

5	**Example--**
	#include <stdio.h>
	void main()
	{
	int **A[5]** = { **-3, 5, 4, 2, 9** };
	int **Sum**=0, i;
	for(i=0; i<5; i++)
	{
	Sum=**Sum** + **A[**i**]**;
	}
	printf("Sum=%d", **Sum**);
	}
	Output
	Sum = 17
	A is an Array variable of Integer type
		Example--
		#include <stdio.h>
		structDate
		{
		int Day;
		char Month[10];
		int Year;
		};
		void main()
		{
		structDated;
		d.Day=**20**;
		strcpy(**d.Month**," **AUGUST**");
		d.Year=**2022**;
		
		printf("Date is **%d**/ **%s**/ **%d**", **d.**Day, **d.**Month, **d.**Year);
		
		}
		Output
		
		Date is **20 / AUGUST / 2022**
		d is a variable of DateStructure datatype

56.

Explain the following

i) Arrays within Structure

ii) Nested Structure

i) Arrays within Structure

A structure contains an array as its data member variable is called Array within structure.

Example:-

```
1. structStudent
{
    int rollno;
    char name[30];
};
```

← Array within Structure

```
2.  struct Employee
{
    int empid;
    char ename[50];
};
```

← Array within Structure

In above **example 1** **Student** is created as User defined data type and **name[30]** is an Array variable which is acting as Data member which is called as **Array with in Structure**.

Similarly in **example 2** **Employee** is created as User defined data type and **ename[50]** is an Array variable which is acting as Data member which is called as **Array with in Structure**.

ii) Nested Structure

A nested structure in C is a **structure within structure**.

One structure can be declared inside another structure in the same way structure members are declared inside a structure.

```
structname_1
{
   member1;
   member2;
     ...
   member n;
};
```

```
structname_2
{
   member1;
   member2;
     .....
struct name_1 var;
   member n;
};
```

← Structure within Structure

A nested structure in C is a structure within structure.

Example:-

```
struct Employee
{
 Int emp_id;
 char name[20];
 int salary;
};
```

```
struct Organisation
{
   char organisation_name[20];
   char org_number[20];
  structEmployee emp;
};
```

← Structure within Structure

57.

Write a C program to display the Palindrome of any given number using recursion.

```
#include <stdio.h>
int reverse(int num); // function prototype
voidmain()
{
   int num;
printf("Enter a number: ");
scanf("%d", &num);
if(num == reverse(num)) // function call
{
printf("%d is a palindrome number.", num);
   }
else
{
printf("%d is not a palindrome number.", num);
   }
 }
// function definitionto reverse the digits of a number
int reverse(int num)
{
  int sum = 0;
if(num != 0)
{
      int remainder = num % 10;
      sum = sum * 10 + remainder;
reverse(num / 10); // recursive call
}
      return sum;
}
```

OUTPUT

```
Enter a number: 7667

7667 is a palindrome number.

Enter a number: 567

567 is not a palindrome number.
```

58.

Write a C program to pass structure variable as function argument.

```
#include <stdio.h>
struct Student // define a structure for a student record
{
    char name[50];
    int rollNo;
    float marks;
};
void displayRecord(struct Student s);// function prototype
void  main()
{
    struct Student s1 = {"John Doe", 101, 85.5};
    displayRecord(s1);// function call
}
void displayRecord(struct Student s) // function definition to display student
{
        printf("Name: %s\n", s.name);
        printf("Roll Number: %d\n", s.rollNo);
        printf("Marks: %.2f\n", s.marks);
}
```

OUTPUT

Name: John Doe

Roll Number: 101

Marks: 85.50

59.

How do you define a structure within a structure? Explain with an example.

Structure within Structure

One structure can be declared inside another structure in the same way structure members are declared inside a structure.

```
structname_1                    structname_2
{                               {
   member1;                        member1;
   member2;                        member2;
     ...                             .....
                                   struct name_1var;
   member n;                       member n;
 } ;                             } ;
```

← Structure within Structure

A nested structure in C is a structure within structure.

Example:-

```
struct Employee
{
  Int emp_id;
  char name[20];
  int salary;
};
```

```
struct Organisation
{
    char organisation_name[20];
    char org_number[20];
    struct Employee emp;
};
```

← Structure within Structure

60.

Describe about union data type used in C.

A union is a user-defined data type that allows you to store different data types in the same memory location.

Like a structure, a union can have multiple members, but only one member can be accessed at a time.

The size of a union is determined by the size of its largest member.

Syntax---

```
union union_name
{
  data_type member1;
  data_type member2;
  ....
  data_type memeberN;
 };
```

C Program

```
union employee
{
  int id;
  char name[50];
 } e1 ;

void main( )
{
e1.id=101;
strcpy( e1.name , "MUTEEB");
union employee e2;
printf( "employee 1 id : %d\n", e1.id );
printf( "employee 1 name : %s\n", e1.name );
}
```

UNIT-V

61.

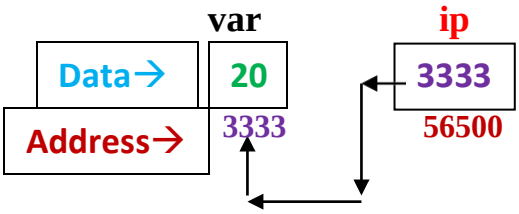
Describe how pointers are declared and initialized in C.

Definition-- A pointer is a variable which stores the address location of another variable in memory.

- Pointers provide *direct access to memory.*
- Pointers allows us to perform dynamic memory allocation and deallocation.
- Pointers helps us to build data structures like *linked list, stack, queues, trees, graphs* etc.

Syntax---

Datatype *Variable;



Example—

```
#include <stdio.h>
void main ()
{
int var = 20;
int *ip;    /* Declaring pointer to an integer */
ip = &var;/* Initializing a pointer variable */
printf("Address stored in ip variable: %u\n",ip);
 }
```

62.

Write a C program that reads and displays the contents of the file.

```
#include  <stdio.h>
#include <stdlib.h>
voidmain()
{
        char ch, file_name[25];
        FILE *fp;
        clrscr();
        fp = fopen("simple.c", "r"); // read mode
        if (fp == NULL)
        {
                printf("file not found.\n");
        }
        printf("The contents of file are:\n");
        while((ch = fgetc(fp)) != EOF)
        printf("%c", ch);
        fclose(fp);
}
```

Simple.c

```
void main()
{
    clrscr();
    printf("Hello World");
     getch();
}
```

Output: The contents of file are:

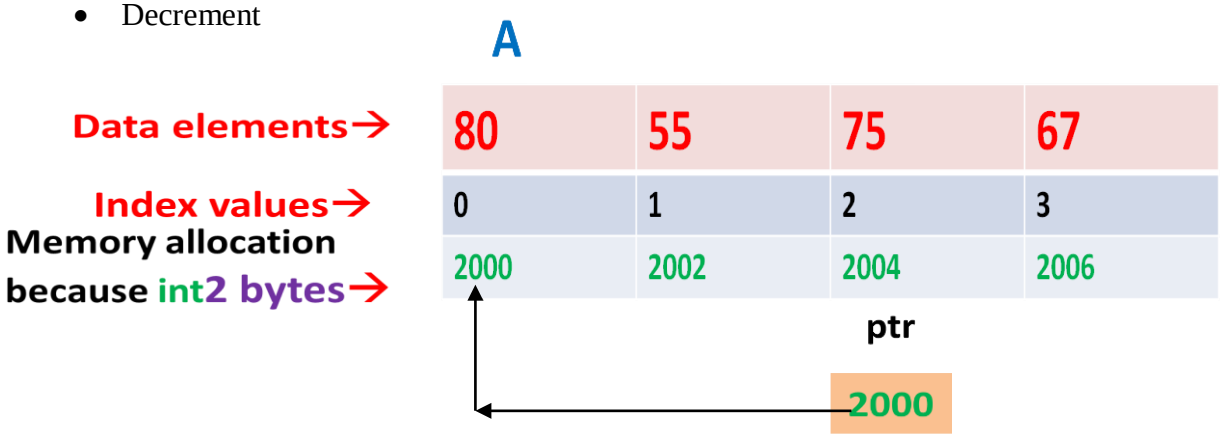
```
void main()
{
clrscr();
printf("Hello World");
getch();
}
```

63.

Explain various arithmetic operations used on pointers.

We can perform arithmetic operations on the pointers like addition, subtraction, etc.
The arithmetic operations on pointers are

- Increment
- Decrement



Increment/Decrement--If we increment a pointer by 1, the pointer will start pointing to the immediate next memory location.
If we decrement a pointer by 1, the pointer will start pointing to the immediate previousmemory location.

```
#include<stdio.h>
void main ()
{
intA[4]={80, 55, 75, 67};
int*ptr;          ptr=&A;                          //ptr has address of A (2000)
printf("Value of *ptr variable: %d\n",*ptr);        // Value at address is 80
ptr ++;           // ptr is incremented which means it will point to next memory location(2002)
printf("\n Value of *ptr variable: %d\n",*ptr);      // Value at address is 55
ptr--;            // ptr is Decremented means it will point to previous memory location(2000)
printf("\n Value of *ptr variable: %d\n",*ptr);        // Value at address is 80
}
```

Output:

```
Value of *ptr variable: 80
Value of *ptr variable: 55
Value of  *ptr variable:80
```

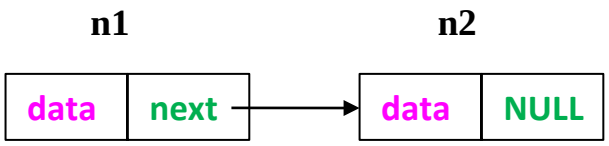
64.

What is self-referential structure and explain its usage.

A structure can have data members which point to a structure variable of the same type.

These types of structures are called self referential structures and are widely used in dynamic data structures **like trees, linked list**, etc.

```
struct node
{
    int data;
struct node*next;
}n1,n2;
```



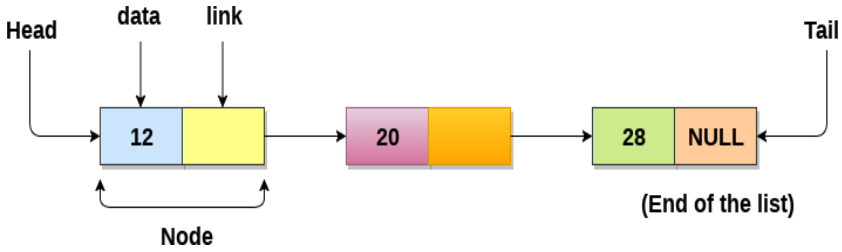
Here, **next** is a pointer to a **struct node** variable.

Example for Self-Referential structure is LINKED LIST----

A linked list is a **linear data structure consisting of elements called nodes** where each node is composed of two parts: an **information** part and a **link** part.

Notion Of Linked List

```
structnode
{
int data;
struct node*next;
};
Struct node*head, *ptr;
```



65.

Write a C program to search text in a file

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

#define MAX_LINE_LENGTH 1000

int main()
{
   char filename[100], search_string[100], line[MAX_LINE_LENGTH];
   int line_number = 0, string_found = 0;

printf("Enter the name of the file to search: ");
scanf("%s", filename);

printf("Enter the string to search: ");
scanf("%s", search_string);

   FILE *fp = fopen(filename, "r");

   while (fgets(line, MAX_LINE_LENGTH, fp))
{
line_number++;
      if (strstr(line, search_string) != NULL)
{
printf("Line %d: %s", line_number, line);
string_found = 1;
      }
   }
```

```
    if (string_found == 0)
{
printf("String not found in file.\n");
    }
fclose(fp);
    return 0;
}
```

66.

Explain array of pointers with an example.

An array of pointers is an array whose elements are pointers to other values or objects. It allows us to create an ordered collection of pointers, where each element in the array points to a different location in memory.

example:

int *array[3];

This declares an array of three integer pointers. Each element in the array can hold a memory address pointing to an integer variable. We can then use these pointers to access or modify the values of the integers they point to.

We can also assign values to each element of the array of pointers, like this:

int a = 1, b = 2, c = 3;

array[0] = &a;

array[1] = &b;

array[2] = &c;

This assigns the addresses of the integer variables a, b, and c to the first, second, and third elements of the array, respectively.

We can then use the array of pointers to access or modify the values of the integers they point to, like this:

***array[0] = 10; // sets the value of a to 10**

***array[1] = 20; // sets the value of b to 20**

***array[2] = 30; // sets the value of c to 30**

we are using **the dereference operator (*) to access the value of the integer that each pointer points to, and then we are assigning a new value to it.**

Array of pointers can be very useful, to create a collection of strings or to implement dynamic data structures such as linked lists.

67.

Explain various File handling functions used in C.

A file is a container in computer storage devices which is used for storing output or information permanently in the form of a sequence of bytes on the disk.

FILE OPERATIONS IN C LANGUAGE

1--Creation of a file (FILE *fptr)
2-- Opening a file(fopen())
3-- Reading a file(fscanf(), fgetc(), fgets())
4--Writing to a file --(fprintf(), fputc(), fputs())
5-Closing a file (fclose())

a)fopen()

fopen()is a standard C library function that is used to open a file based on the mode of operation like read, write and append.

Syntax---

FILE ***fopen**(const char ***filename**, const char ***mode**)

b)fprintf()

fprintf()is a standard C library function that is used to write set of characters into a file.

Syntax---

int **fprintf**(FILE ***stream**, const char ***format**, ...)

c)fputs()

fputs() is a standard C library function that is used to write a string of characters to a file at the location indicated by the file pointer.

Syntax-

int **fputs**(const char ***str**, FILE ***stream**)

d)fscanf()

The fscanf() function is used to read set of characters from file. It reads a word from the file and returns EOF at the end of file.

Syntax---

int **fscanf**(FILE ***stream**, const char ***format [, argument, ...]**)

68.

Write a C program to count number of lines, words and characters in a given file.

```
#include  <stdio.h>
#include <stdlib.h>
void main()
{
        FILE *file;
        char ch;
        int characters, words, lines;
        file = fopen("Simple.txt", "r");
        if (file == NULL)
        {
                printf("\n File not found.");
        }
        //Logic to count characters, words and lines.
        characters = words = lines = 0;
        while ((ch = fgetc(file))!=EOF)
        {
                characters++;
                if (ch == '\n' || ch == '\0')// Check new line
                        lines++;
                if (ch == ' ' || ch == '\t' || ch == '\n' || ch == '\0')// Check words
                        words++;
        }
        if (characters > 0)
        {
                words++;
                lines++;
        }
        /* Print file statistics */
        printf("\n Total characters = %d\n", characters);
        printf("Total words      = %d\n", words);
        printf("Total lines      = %d\n", lines);
        fclose(file);
}
```

Simple.txt

```
void main()
{
clrscr();
printf("Hello World");
getch();
}
```

Output:

```
Total characters =57
Total words      = 8
Total lines      = 6
```

69.

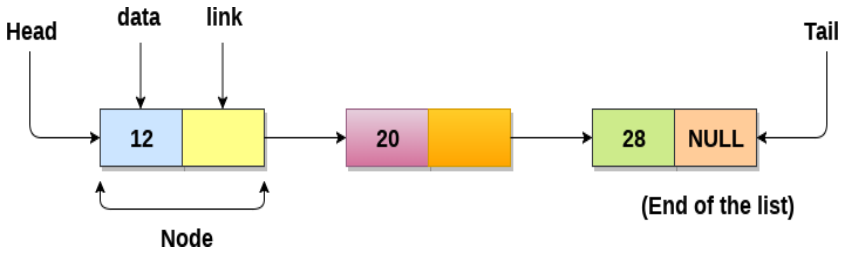
Describe Linked List with its applications.

LINKED LIST

A linked list is a **linear data structure consisting of elements called nodes** where each node is composed of two parts:
an **information** part and a **link** part, also called the next pointer part.

Notion Of Linked List

```
structnode
{
int data;
structnode*next;
};
structnode*head, *ptr;
```



APPLICATIONS OF LINKED LIST

- Implementation of **stacks** and **queues**
- Implementation of **graphs**
- **Dynamic** memory allocation and deallocation.
- Manipulation of polynomials
- ***Image viewer***
- ***Previous and next page in web browser***
- ***Music Playerwith previous and next Songs playing list***

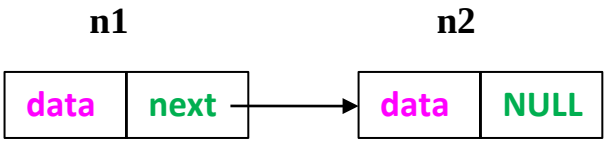
70.

Explain about self-referential structure with an example.

A structure can have data members which point to a structure variable of the same type.

These types of structures are called self referential structures and are widely used in dynamic data structures **like trees, linked list**, etc.

```
struct node
{
    int data;
structnode*next;
}n1,n2;
```



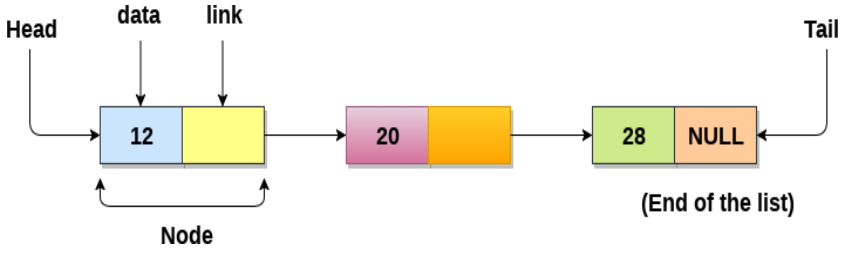
Here, **next** is a pointer to a **struct node** variable.

Example for Self-Referential structure is LINKED LIST----

A linked list is a **linear data structure consisting of elements called nodes** where each node is composed of two parts: an **information** part and a **link** part.

Notion Of Linked List

```
structnode
{
int data;
structnode*next;
};
structnode*head, *ptr;
```



71.

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

```
#include  <stdio.h>
#include <stdlib.h>
void main()
{
        char ch;
        FILE *source_file, *target_file;
        printf("Enter name of file to copy\n");
        source_file=fopen("simple.c", "r");
target_file = fopen("new.txt", "w");

if(source_file == NULL )
        {
                printf("File Not Found\n");
        }
        if(target_file == NULL )
        {
                fclose(source);
                printf("File not found to copy\n");
        }
        while( (ch = fgetc(source) ) != EOF )
                fputc(ch, target);
        printf("File copied successfully.\n");
        fclose(source);
        fclose(target);
}
```

simple.c

```
void main()
{
   clrscr();
printf("Hello World");
   getch();
}
```

new.txt

```
void main()
{
   clrscr();
printf("Hello World");
   getch();
}
```

Output:

File copied successfully

72.

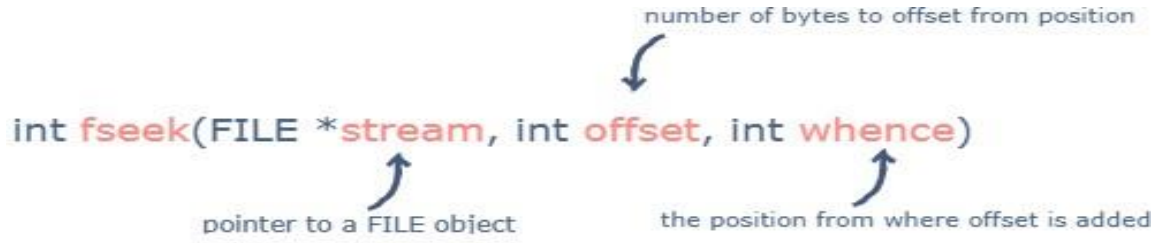
Explain the following functions used in File operations

a)fseek() b) fputs() c) fopen()

a) fseek()

fseek() is a standard C library **function which sets the file position of the stream to a given offset.** The pointer associated with the file is moved to that offset.

Syntax-



whence defines the point with respect to where the file pointer needs to be moved. It is specified by one of the following constants:

- **SEEK_END**: End of the file.
- **SEEK_SET**: Beginning of the file.
- **SEEK_CUR**: Current position of the file pointer.

b) fputs()

fputs() is a standard C library **function that is used to write a string of characters to a file at the location indicated by the file pointer.**

Syntax-

int **fputs**(const char ***str**, FILE ***stream**)

str---- String of Characters to be written.
stream ----- Pointer to the file where the character is to be written.

c) fopen()

fopen()is a standard C library **function that is used to open a file based on the mode of operation like read, write and append.**

Syntax---

FILE ***fopen**(const char ***filename**, const char ***mode**)

filename -- This represents the string which is a file name like one.txt , etc.
mode ------- This represents the read or write or append the data to/from a file.

73.

Explain the following functions used in File operations

a) fprintf() b) rewind() c) fscanf()

a) fprintf()

fprintf()is a standard C library **function that is used to write set of characters into a file.**

Syntax---

int **fprintf**(FILE ***stream**, const char ***format**, **...**)

stream--- this is the pointer to a FILE object that identifies the stream.
format --- This is the string that contains the text to be written to the stream.

b) rewind()

The rewind() **function sets the file pointer at the beginning of the stream**. It is useful if you have to use stream many times.

Syntax---

void **rewind**(FILE ***stream**)

stream -- The stream whose file position indicator is to be set to the beginning of the file.

c) fscanf()

The fscanf() **function is used to read set of characters from file**. It reads a word from the file and returns EOF at the end of file.

Syntax---

int **fscanf**(FILE ***stream**, const char ***format [**, argument, ...**]**)

stream -- This is the pointer to a FILE object that identifies the stream.
format --- This is the string that contains the text to be read using format specifier to the stream.

74.

Describe about different modes of File operations used in C?

In C, there are several modes of file operations that can be used to open and work with files.

These modes determine how the file is accessed, whether the file is created if it does not exist, and whether the file can be modified.

"r" mode -This mode is used to read from an existing file. If the file does not exist, it returns an error.

"w"mode - This mode is used to write to a file. If the file does not exist, it is created. If the file already exists, it is truncated (its contents are deleted) and a new file is created.

"a"mode -This mode is used to append to an existing file. If the file does not exist, it is created. If the file already exists, data is added to the end of the file without overwriting the existing content.

"r+" mode -This mode is used to read from and write to an existing file. If the file does not exist, it returns an error.

"w+"mode -This mode is used to read from and write to a file. If the file does not exist, it is created. If the file already exists, it is truncated (its contents are deleted) and a new file is created.

"a+"mode -This mode is used to read from and append to an existing file. If the file does not exist, it is created. If the file already exists, data is added to the end of the file without overwriting the existing content**.**

75.

How do pointers passed as arguments to a function in C? And explain it.

In C, a pointer is a variable that holds the memory address of another variable.

When you pass a pointer as an argument to a function, you are passing a reference to that memory address.

This allows the function to access and modify the data stored at that memory location, even if it is outside of the function's own scope.

To pass a pointer as an argument to a function, you can declare the function parameter as a pointer type, and then pass the address of the variable you want to pass as an argument using the "address of" operator "&".

For example:

```
void myFunction(int *ptr)
{
   *ptr = 100; // set the value pointed to by ptr to 100
}
int main()
{
    int Var = 42;
    printf("Before Var = %d\n", Var);
    myFunction(&Var); // pass the address of myVariable to myFunction
     printf("After Var = %d\n", Var);
   return 0;
}
```