C-PROGRAMMING 2071 QUESTION SET

1. a) What is mean by compilation? What is meant by interpretation? How do these two process differ?

Ans- Compilation is the process of converting source code of programming language into machine code.

Interpretation is the process of reading and interpreting the code line by line executing each instruction in real time.

Compilation	Interpretation
1. Takes entire program as an input.	1. Takes single instruction at a time.
2. Less execution time.	2. More execution time.
3. More memory requirement due to object code generated.	3. Less memory requirement since no intermediate code is generated.

b) Define programming language? What are the feature of good programming language?

Ans- Programming language is a formal language used to communicate with the computer in the form of instruction.

Features of good programming language:

- A programming language must be simple, easy to learn and use, have good readability, and be human recognizable.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and its execution consumes little space in memory.
- A programming language should be well structured and documented so that it is suitable for application development.
- Necessary tools for the development, debugging, testing, maintenance of a program must be provided by a programming language.

2. a) What is preprocessor directives? Explain constants and variables.

Ans- Preprocessor directives are line that starts with character # and used to make source programs easy to change and easy to compile in different execution environments.

Constants: Its value is fixed throughout the program . Constants provide a way to use specific, predefined values in the code without the risk of accidentally modifying them. Constants can be of different data types, such as integers, floating-point numbers, characters, or strings. In C, we can define constants using various methods, including preprocessor **#define** directives, **const** keyword, and enumerations. Eg: const data_type CONSTANT_NAME = constant_value;

Variables: In C programming, a variable is a named location in memory that stores a value. It is a fundamental concept used to represent and manipulate data within a program. Variables have a data type, which defines the type of data they can hold, such as integers, floating-point numbers, characters, or user-defined types.To use a variable in C, you must declare it before you can assign a value to it or use its value in expressions. Eg: data_type variable_name;

b) Write the syntax and example of following statement and function:

I. Printf(): It's function is to send formatted string to standard output.
   Syntax: printf("format string",argument_list);
   Example: printf("my name is %s",name);

II. Scanf(): It's function is to convert text stream coming from keyboard to data value and store them in program variable.
   Syntax: int scanf(const char *format, ...);
   Example: scanf("%s",name);

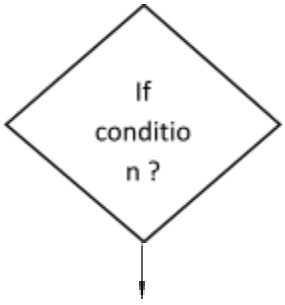
III. Getch(): It's function is to get a character from keyboard but doesn't echoes to the screen.
   Syntax: character_variable=getch();
   Example: name=getch();

IV. Long(): It is used to represent signed integer value with large range than int datatype.
   Syntax: long variableName;
   Example: long number=9843510038;

3. a) What is control statement? Illustrate nested IF statement with it's flowchart. Write a program to calculate the series: 1*10+3*20+6*30+……..+ $\frac{N(N+1)}{2}$*10N, where N is integer term read from the keyboard.

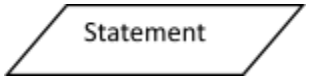
Ans- Control statement are those statement which are used for decision making and executing different type of code blocks based on certain condition.

Syntax of IF statement: if (condition) {
statement; }




```
#include<stdio.h>

int main()    {
```




```
int n,i,a=0,sum=0;
printf("enter a number");
scanf("%d",&n);
for(i=1;i<=n; i++)          {
    sum=sum+a;
    printf("%d\n",a);
    a=i*(i+1)/2*(10*i);       }

printf("sum is %d",sum);
return 0;        }
```

Output: enter a number 3

0 10 60

Sum is 70

4. **a) Write a program to display Armstrong number between the range enter by the user and also display their counts. You must use a function to check for Armstrong numbers and display them from main.**

Ans-

```
#include <stdio.h>
#include <math.h>
int armstrong(int);
int main()    {
        int start, end, temp, count;
        printf("enter range: ");
        scanf("%d%d", &start, &end);
        if (start > end)    {
                temp = start;
                start = end;
```

```
            end = temp;     }
        printf("the armstrong numbers between %d and %d are:\n", start, end);
        for (count = start; count <= end; count++)      {
            if (count == armstrong(count))    {
                printf("%d is an armstrong number\n", count);     }    }
return 0;     }
int armstrong(int n)    {
        int temp = n, numDigits = 0;
        while (temp != 0)      {
            temp = temp / 10;
            numDigits++;     }
        temp = n;
        int sum = 0;
        while (temp != 0)   {
            int rem = temp % 10;
            sum += pow(rem, numDigits);
            temp = temp / 10;    }
    return sum;       }
```

b) What do you mean by nested function and recursive function? Give an example of recursive function.

Ans- Sure, here are simple definitions for nested and recursive functions:

1. Nested Function:

A nested function is a function that is defined within another function. It is enclosed within the scope of its containing function and can access the variables and resources of its parent function. Nested

functions are often used for organization and encapsulation, allowing you to group related functionality together.

2. Recursive Function:

A recursive function is a function that calls itself in order to solve a problem by breaking it down into smaller instances of the same problem. Recursion involves solving a base case directly and then using the function to solve progressively simpler versions of the problem until the base case is reached. Recursive functions are commonly used for tasks that exhibit self-similar or repetitive structures.

```
#include<stdio.h>

int fibo(int num);

int main()

{

   int num,i;

   printf("enter a number");

   scanf("%d",&num);

   for(i=1;i<num;i++)

   {

      printf("%d",fibo(i));

   }

}

int fibo(int num)

{

   if(num==0 || num==1)

   return num;

   else

   return (fibo(num-1)+ fibo(num-2));

}
```

Output: enter a number 9

112358

5 a) write a C program to read string and display its reverse. Use user defined function to count number of characters in it and to reverse it.

```
#include <stdio.h>

#include <string.h>

int countCharacters(char[]);

void reverseString(char[]);


int main()     {

   char inputString[100];

   printf("Enter a string: ");

   fgets(inputString, sizeof(inputString), stdin);       // Remove the newline character from the input

   if (inputString[strlen(inputString) - 1] == '\n')

   {      inputString[strlen(inputString) - 1] = '\0';    }

  int charCount = countCharacters(inputString);

  printf("Number of characters in the string: %d\n", charCount);

  printf("Original string: %s\n", inputString);

   reverseString(inputString);

   printf("Reversed string: %s\n", inputString);

   return 0;

}

int countCharacters(char str[])

{

   int count = 0;
```

```
    while (str[count] != '\0')
    {    count++;    }
    return count;    }
void reverseString(char str[])
{
    int length = strlen(str);
    for (int i = 0; i < length / 2; i++)
    {
        char temp = str[i];
        str[i] = str[length - i - 1];
        str[length - i - 1] = temp;
    }
}
```

b) Write a algorithm to insert a value in a array at a position given by user.

Ans:

1. Read the size of the array n
2. Declare an array arr of size n
3. Read the elements of the array arr from the user
4. Read the value to be inserted (newValue)
5. Read the position where the value should be inserted (insertPosition)
6. If insertPosition is less than 0 or greater than n, then
   Print "Invalid position"
   Exit
7. Shift the elements of the array to the right from index n-1 to insertPosition:
   a. Start from the last element (i = n-1)
   b. Repeat while i is greater than or equal to insertPosition:

arr[i+1] = arr[i]

Decrement i by 1

8. Insert the newValue at the specified insertPosition:

arr[insertPosition] = newValue

9. Increment n by 1 to reflect the new size of the array

10. Print "Array after insertion:"

11. Loop i from 0 to n-1:

Print arr[i] followed by a space

12. End

6 a) What is tag? Must a tag be included in a structure type definition? Must a tag be included in a structure variable declaration? Explain.

Ans**:** In C programming, a tag is a name given to a structure. It is used to define a new user-defined data type using the struct keyword. A tag is essentially the name of the structure itself. It allows you to create instances of the structure and use it to store related data fields together.

No, a tag is not strictly required in a structure type definition, but it's a good practice to include one. When you provide a tag name, you can later use that name to declare variables of that structure type, making your code more organized and understandable.

Yes, if you want to declare a variable of a structure type, you must include the tag. The tag is used to specify the type of the variable.

b) Write a C program that reads several different names and address using structure computer, rearrange the names into alphabetical order and write out alphabetical list.

Ans- #include<stdio.h>

```
int main()

{

   struct computer

   {

      char name[20];

      char address[20];
```

```
  }s[3];
  char temp[10];
for(int i=0;i<3;i++)
  {
     printf("enter name and address");
     scanf("%s%s",s[i].name,s[i].address);
  }
  for(int i=0;i<3;i++)
  {
     for(int j=i+1;j<3;j++)
     {
        if(strcmp(s[i].name,s[j].name)>0)
        {
           strcpy(temp,s[i].name);
           strcpy(s[i].name,s[j].name);
           strcpy(s[j].name,temp);

           strcpy(temp,s[i].address);
           strcpy(s[i].address,s[j].address);
           strcpy(s[j].address,temp);
        } }
     printf("%s%s",s[i].name,s[i].address);
    }
  return 0;
}
```

7) Illustrate with example that "Array is indirectly a pointer". Write program to calculate the sum and average of a integer number between M and N (where value of M and N are read from keyboard) using pointer.

Ans- In C and C++, an array is essentially a contiguous block of memory that stores elements of the same data type. When you declare an array, you're actually creating a pointer that points to the first element of the array. Let's illustrate this with an example:

```
#include<stdio.h>

int main()

{

   float x[5], sum=0.0, avg;

   int i;

   float *px, *psum, *pavg;

   px = &x[0];  //  Or, px = &x;

   psum = &sum, pavg = &avg;

   printf("Enter array Elements: ");

   for (i=0;i<5;i++)

   {

      scanf("%f",(px+i));

      *psum += *(px + i);

   }


   *pavg = *psum / 5;

   printf("Sum= %.2f \t Average= %.2f\n", *psum, *pavg);


   return 0;
```

}

Enter array Elements: 10
24
30
62
15
Sum=141.00 Average=28.20

8) Write a program to continuously read name, age and salary of a worker and write it into a file until user confirms to end. Then read n from user and display the nth record in the file. Details of worker must be represented by a structure.

Ans- #include <stdio.h>

#include <string.h>

typedef struct

{

   char name[30];

   int age;

   int salary;

} worker;

int main()

{

   worker w[50];

   FILE *p;

   int i = 0, n;

   char answer[5];

```
p = fopen("worker.txt", "wb+");
if (p == NULL)
{
   printf("File Error!");
   return 1;
}

do
{
   printf("Enter name, age and salary:\n");
   scanf("%s %d %d", w[i].name, &w[i].age, &w[i].salary);
   fwrite(&w[i], sizeof(w[i]), 1, p);
   printf("Store another record? (yes or no):\n");
   scanf("%s", answer);
   i++;
} while (strcmp(answer, "no") != 0);

rewind(p);

printf("Enter which record to display?\n");
scanf("%d", &n);

for (i = 0; i < n; i++)
{
   fread(&w[i], sizeof(w[i]), 1, p);
```

```
    }

    printf("Worker Name: %s\n", w[n - 1].name);
    printf("Worker Age: %d\n", w[n - 1].age);
    printf("Worker Salary: %d\n", w[n - 1].salary);
    fclose(p); // Don't forget to close the file
    return 0;
}
```

Output: Enter name, age and salary:

John 25 50000

Store another record? (yes or no):

yes

Enter name, age and salary:

Jane 30 60000

Store another record? (yes or no):

no

Enter which record to display?

2

Worker Name: Jane

Worker Age: 30

Worker Salary: 60000

9 a) Compare arithmetic and logical if statement in FORTRAN.

Ans- **The arithmetic IF statement is used to select one of several alternatives based on a given arithmetic expression. It's typically used when you have multiple ranges of values and you want to execute different blocks of code based on which range the expression falls into.**

Syntax: IF (condition) label1, label2, label3

The logical IF statement is used to conditionally execute a single statement or a block of statements based on a logical expression (true or false).

Syntax: IF (logical_expression) statement

b)Write a FORTRAN program to display nature of roots of a quadratic equation. Calculate and display the roots, if they are real and equal.

```
PROGRAM QuadraticRoots

 IMPLICIT NONE


 REAL :: a, b, c, discriminant, root1, root2


 ! Read coefficients from user

 WRITE(*,*) "Enter coefficients of quadratic equation (a, b, c):"

 READ(*,*) a, b, c


 ! Calculate discriminant

 discriminant = b**2 - 4*a*c


 ! Check nature of roots

 IF (discriminant > 0.0) THEN

   WRITE(*,*) "Roots are real and distinct."

   root1 = (-b + SQRT(discriminant)) / (2*a)

   root2 = (-b - SQRT(discriminant)) / (2*a)

   WRITE(*,*) "Root 1 =", root1

   WRITE(*,*) "Root 2 =", root2

 ELSE IF (discriminant == 0.0) THEN

   WRITE(*,*) "Roots are real and equal."
```

```
    root1 = -b / (2*a)
    WRITE(*,*) "Roots =", root1
  ELSE
    WRITE(*,*) "Roots are complex."
  END IF
END PROGRAM QuadraticRoots
```