Set - 2019

1.a. Define the following terms: C tokens, keywords, Identifiers. Discuss different data types in C with examples. Why do we comment a program? [6+4+2]

ans. In C programming, a token is the smallest unit of a program that is meaningful to the compiler. Tokens are the building blocks of a C program and are used to create statements, expressions, and other program elements. There are six types of token in C programming:

1. keywords
2. Identifiers
3. Constants
4. strings
5. Operators
6. Special characters.

1. Keywords:

Keywords are the reserved words in the program that have a predefined meaning and cannot be used as variable names or identifiers. Keywords are written in lower case. Examples of keywords include if, else, while, for, return, switch etc.

2. Identifiers:

Identifiers also known as variable is a container to store the data. It is a name given to the memory location where the actual data is stored. Variable names may have different data types

to identify the type of value stored. An identifier must start with a letter or an underscore and can contain letters, digits, and underscores. For eg:

int a ; this means that a can store only one integer.

3. Constants:

Constants are fixed values that do not change during program execution. There are several types of constants in C, including integer constants, character constants, floating-point constants, and string literals. It contains two type: literal & symbolic constant.

i. Literal Constant: A literal constant is value type directly into computer.

ii. Symbolic constant: A symbolic constant is the name just like a variable initialized with const. Integer, constants, character constants, floating-point constants ~~& string literals~~ lies under symbolic constant.

Syntax:

<const><data type><identifiers><const value>

4. Strings:

Strings are a sequence of characters enclosed in double quotes. They are used to represent text data in a C program.

5 Operators:

Operators are symbols used to perform mathematical or logical operations on operands. Examples of operators in C include arithmetic operators (+, -, *, /), relational operators (==, !=, <, >, <=, >=), and logical operators (&&, ||, !).

6. Special characters:

Special characters are symbols used to represent special functions in a C program. Examples of special characters include parentheses ( ), braces { }, square brackets [ ], commas (,), semicolons (;), and the period (.) used for accessing structure members.

Tokens are essential for constructing a valid and meaningful C program, and understanding their types & uses is important for any C programmer.

- [2021 Q.No. 10] → Data types in C

- Commenting a program is an important aspect of programming practice, as it helps to improve code readability, maintainability, and collaboration. Here are some reasons why we comment a program in C:

1. Code Readability:

Comments can help make code more readable by explaining the purpose and function of individual lines or blocks of code. This can be especially

helpful for complex or lengthy programs that might be difficult to understand at first glance.

2. Code maintenance:

Comments can also be helpful when maintaining or update existing code. They can provide information about why certain decisions were made or how specific algorithms work, which can make it easier to modify or improve the code in the future.

3. Collaboration:

Comments can help facilitate collaboration among team members working on the same codebase. By explaining the intent and purpose of certain sections of code, comments can make it easier for others to contribute and provide feedback.

4. Documentation:

Comments can serve as a form of documentation for a program, providing an overview of how the code works and what its key features are. This can be especially useful for larger projects or programs that may be used by others outside of the original development team.

By taking the time to add comments to your code, you can make it easier to understand & use for yourself & others.

2.a. What is the difference between while loop and do-while loop? Find the output that will be generated by the following program. [2+4]

ans.

S.No.	While loop	Do-While loop
1.	While loop is a pre-test loop.	Do-while loop is a post-test loop.
2.	It test the condition first before executing the loop body.	It test the condition at the end of the loop body.
3.	The body of the loop may or may not be executed at all.	The body of the loop will be executed at least once because the condition is checked last.

4.	Syntax:
	while (condition)
	{
	statements;
	}
		Syntax:
		do
		{
		statements
		} while (expression);

5.	No semicolon at the end of the condition	A semicolon at the end of the condition.

// Find the output

```
#include<stdio.h>
int main()
{
    int i=0,x=0;
    while(i<20){
        if(i%5==0){
            x+=i;
            printf("%d",x);
        }
        ++i;
    }
    printf("\nx=%d",x);
}
```

Output:

0 5 15 30
30

b. What is the difference between '=' and '==' operators in C. Write a C program that asks user to enter temperature in Celsius (C) and converts it in Fahrenheit(F). The conversion formula is C:C=5(F-32/9): [2+4]

S.No.	'=' operator	'==' operator
1.	'=' is an assignment operator.	'==' is a comparison operator.
2.	It is used to assign value to a variable.	It is used to compare two variable.

3.	Example:
	int x = 10;
		Example:
		int x = 10;
		if (x == 5){
		statements;
		}

```
// Converting celsius into Fahrenheit.
#include<stdio.h>
int main( )
{
    float cel=0, fah=0;
    printf("\n Provide temperature in celsius: ");
    scanf("%f", &cel);
    fah = (cel*9 + 32)/5;
    printf("\n The fahrenheit is %.1f ", fah);
    return 0;
}
```

3a. What do you mean by precedence and associativity of operators? A marking scheme in a particular exam is as follows:

Marks (in %)	Grade
80 - 100	A
60 - 79	B
50 - 59	C
40 - 49	D
0 - 39	E

Write a program that asks student to enter his/her marks in any subject and print the equivalent grade. Your program should display "Invalid Input!", if the mark entered is less than 0 or more than 100. [4+8]

ans. In programming, the precedence of an operator refers to the order in which it is evaluated in an expression with multiple operators. Operators with higher precedence are evaluated first.

For example, in the expression '2+3*4', the multiplication operator (*) has a higher precedence than the additional operator (+), so, the multiplication is evaluated first, resulting in '2+12', which equals 14.

Associativity, on the other hand, refers to the direction in which operators are evaluated when they have the same precedence. Operators can be left-associative or right-associative. Left-associative operators are evaluated

from left to right, while right-associative operators are evaluated from right to left.

For example, the assignment operator ('=') is right associative, meaning that in an expression like 'a = b = c', the value of 'c' is assigned to 'b' first, and then the value of 'b' is assigned to 'a'.

A list of operator with their precedence & associativity is given below.

category	operator	Associativity
postfix	( ) [ ] ->	Left to Right
unary	++ --	Right to left
multiplicative	* / %	Left to Right
Additive	+ -	Left to Right
Relational	< <= > >=	Left to Right
Equality	== !=	Left to Right
Logical AND	&&	Left to Right
Logical OR	||	Left to Right
conditional	?:	Right to left
Assignment	=, +=, -=, *=, /=,	Right to left
comma	,	left to Right

Example :

1. y = 5 % 2 * 4
   = 1 * 4
   = 4
   ∴ y = 4

```
// A marking scheme
#include<stdio.h>
int main( )
{
    float marks = 0;
    flag:
    printf("\nEnter your marks in any subject you got : ");
    scanf("%f", &marks);
    if(marks>100 || marks<0)
    {
        printf("\nInvalid Input. Please try again.");
        goto flag;
    }
    else
    {
        if(marks>=80 && marks<=100)
        {
            printf("\n You got A in your subject.");
        }
        else if(marks>=60 && marks<=79)
        {
            printf("\nYou got B in your subject.");
        }
        else if (marks>=50 && marks<=59)
        {
            printf("\nYou got C in your subject.");
        }
        else if(marks>=40 && marks<=49){
            printf("\nYou got D in your subject.");
    }
```

```
        else
        {
            printf("\nYou got E. Please study hard.");
        }
    }
    return 0;
}
```

4. Write a program to enter n numbers in an array, and compute the sum of only prime number in the array. [8].

```
#include<stdio.h>
int main()
{
    int num[100], sum=0, count=0, n=0, i=0, j=0;
    printf("\nEnter how many number you want: ");
    scanf("%d",&n);
    for(i=0; i<n ; i++)
    {
        printf("\nEnter number for %d: ",i+1);
        scanf("%d", &num[i]);
        for(j=1 ; j<num ; j++)
        {
            if(num[i]%j==0)
            {
                count++;
            }
        }
    }
```

```
        if(count == 2)
        {
            sum = sum + num[i];
        }
        count = 0;
    }
    printf("\nThe sum of prime in an array is %d", sum);
    return 0;
}
```

5. Write a program to multiply given two matrices of given order.[8]

```
#include<stdio.h>
int main()
{
    int m=0, n=0, p=0, q=0, i=0, j=0, k=0;
    int A[10][10], B[10][10], mul[10][10], total=0;
    printf("\nEnter your elements of row and column of
        first array: \n");
    scanf("%d %d", &m, &n);
    printf("\nEnter your elements by row: \n");
    for(i=0; i<m; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d", &A[i][j]);
        }
    }
```

```
Flag:
printf("\nEnter your elements of row and column
    of second matrix: \n");
scanf("%d %d", &p, &q);
if(n != p)
{
    printf("\nThis matrix cannot be multiply. Try again.");
    goto flag;
}
else
{
    printf("\nEnter your element for 2nd matrix: \n");
    for(i=0; i<p; i++)
    {
        for(j=0; j<q; j++)
        {
            scanf("%d", &B[i][j]);
        }
    }
}
for(i=0; i<m; i++)
{
    for(j=0; j<q; j++)
    {
        for(k=0; k<n; k++)
        {
            total = total + A[i][k] * B[k][j];
        }
        mul[i][j] = total;
    }
}
```

```
printf("\nTheir multification is: \n");
for(i=0; i<m; i++)
{
    for(j=0; j<q; j++)
    {
        printf("%d\t", mul[i][j]);
    }
    printf("\n");
}
return 0;
}
```

7. What do you mean by recursion? Write a program to calculate factorial of a given number wing recursion. [2+6].

ans. If a function calls itself again and again till certain condition meets then this phenomenon is known as recursion in c programming.

Syntax:

```
void recursion( ){
    recursion( ); //function call
}
int main( ){
    recursion( );
}
```

```
#include<stdio.h>
int fact(int);
int main( )
{
    int num = 0 , i = 0 , result = 0;
    printf("\n Enter your number to find factorial: ");
    scanf("%d", &num);
    result = fact(num);
    printf("\n The factorial of %d is %d.", num, result);
    return 0;
}
int fact(int m)
{
    if(m == 1)
    {
        return 1;
    }
    else
    {
        return m * fact(m-1);
    }
}
```

6. What do you mean by functions in C? Write its advantages. Using user defined function, Write a program to find maximum value among three integers. [2+2+4]

ans. In C, a function is a block of code that performs a specific task. It is a reusable piece of code that can be called from different parts of a program. Function in C help to break down a large program into smaller and more manageable parts, making it easier to understand, test, and maintain.

Syntax:

```
#include<stdio.h>
Function declaration;
int main( )
{
    ..... ;
    function call;
    return 0;
}
Function definition
{
    ..... ;
}
```

Some advantages of function are:

1. Reusability:

Functions can be reused in different parts of a program, reducing the amount of code duplication and

making the code easier to maintain.

2. Modularity:

By breaking a program into smaller functions, it becomes easier to understand and test each part separately.

3. Abstraction:

Functions provide a way to hide the complexity of the code by providing a high-level interface to the user.

4. Efficiency:

Functions can help to improve the performance of a program by allowing frequently used code to be optimized and compiled separately.

5. Error handling:

Functions can help to simplify error handling by providing a centralized place to handle errors, rather than having to handle them in multiple places throughout the code.

```
// To find maximum value using function
#include<stdio.h>
int max(int, int, int);
int main( )
{
    int a=0, b=0, c=0, gt=0;
    printf("\nEnter any 3 numbers: ");
    scanf("%d %d %d", &a, &b, &c);
    gt=max(a, b, c);
    printf("\nThe greatest among them is %d", gt);
    return 0;
}
int max(int x, int y, int z)
{
    if(x>y)
    {
        if(x>z)
        {
            return x;
        }
        else
        {
            return z;
        }
    }
    else
    {
        if(y>z)
        {
            return y;
        }
```

```
        else
        {
            return 2;
        }
    }
}
```

8. Write any two differences between structure and union. Describe nested structure with example. [2+6]

ans. Set 2022 [Q.No.8] - Difference between

A nested structure in C ~~has~~ is a structure that has one or more members that are themselves structures. This means that the members of a structure can be of any data type, including other structures.

Syntax:

```
struct structure2 {
    .....
    .....
};
struct structure1 {
    .....
    .....
    struct structure2 {name};
};
```

For example:

```
#include<stdio.h>
struct student{
    char name[50];
    int id;
};
struct college{
    char org_name[100];
    char faculty[10];
    struct student stu[50];
}apple;
int main()
{
    int i=0, j=0, num=0;
    printf("\nEnter name of your organization: ");
    fflush(stdin);
    gets(apple.org_name);
    printf("\nEnter your Faculty name: ");
    fflush(stdin);
    gets(apple.faculty);
    printf("\nHow many students do you have: ");
    scanf("%d", &num);
    for(i=0; i<num; i++)
    {
        printf("\nEnter details for student %d:", i+1);
        printf("\nName: ");
        fflush(stdin);
        gets(apple.stu[i].name);
```

```
        printf("\n Id : ");
        scanf("%d", &apple.stu[i].id);
        printf("\n");
    }
    printf("\n %s Organization ", apple.org_name);
    printf("\n %s Faculty ", apple.faculty);
    for(i=0; i<num; i++)
    {
        printf("\n Details of student %d", i+1);
        printf("\n Name: %s", apple.stu[i].name);
        printf("\n Id: %d", apple.stu[i].id);
    }
    return 0;
}
```

9. Discuss different file opening modes. Write a program to copy content of one text file to another. [4+4]

ans. In C, when working with files, we can specify the mode in which we want to open a file using the 'fopen()' function. The mode is specified as a string argument in the second parameter of the function call. Here are the different file opening modes in C:

a. "r":

This mode opens a file in read only mode. The file must exist, and the file pointer will be positioned at the beginning of the file.

2. "w":

This mode opens a file in write-only mode. If the file already exists, its contents will be truncated to zero length. If the file does not exist, it will be created.

3. "a":

This mode opens a file in append mode. If the file already exists, the file pointer will be positioned at the end of the file, and any new data written to the file will be appended to the existing data. If the file does not exist, it will be created.

4. "r+":

This mode opens a file in read-write mode. The file must exist, and the file pointer will be positioned at the beginning of the file.

5. "wt":

This mode opens a file in read-write mode. If the file already exists, its contents will be truncated to zero length. If the file does not exist, it will be created.

6. "a+":

This mode opens a file in read-write mode. If the file already exists, the file pointer will be positioned at the end of the file, and any new data written to the file will be appended to the existing data. If the file does not exist it will

be created.

In addition to these basic file opening modes, we can also use the "b" (binary) flag to specify that we are working with binary data. For example, "rb" opens a file in binary read-only mode, and "wb" opens a file in binary write-only mode.

10. Write a program using pointer to search a given element in an array of size, N. [8]

```
#include <stdio.h>
int main()
{
    int *ptr, *p, a[100], i=0, num=0, search=0, count=0;
    int hero=0;
    ptr = a;
    p = &search;
    printf("\nHow many number you want in an array:");
    scanf("%d", &num);
    printf("\nEnter your elements:\n");
    for(i=0; i<num; i++)
    {
        scanf("%d", ptr+i);
    }
    flag:
    printf("\nWhich number are you searching for:");
    scanf("%d", p);
    for(i=0; i<num; i++)
    {
        if(*(ptr+i)==*p)
        {
            count++;
            if(count==1)
            {
                printf("\nYour given element is in %d
                    index of array.", i);
            }
```

```
            else
            {
                printf("\n Your given element is again found
                        in %d index of array.", i);
            }
        }
        else
        {
            hero++;
        }
    }
    if(hero==num)
    {
        printf("\n Sorry! Provided elements is not found. Try
                again.");
        goto Flag;
    }
    return 0;
}
```

11. Write short notes on any TWO: [4+4]

a. Algorithms and Flowchart

b. String Handling Functions

c. DMA

ans. Set [2022] Q.No. 11- (a, b, c)