**Q.NO.1**

**a. Define the UNIX Operating System.**

Ans:- The UNIX Operating System is a family of operating systems that originated in the late 1960s at AT&T Bell Labs. It is characterized by its design philosophy of small, simple tools that can be combined to perform complex tasks. UNIX provides a multi-user, multitasking environment and is known for its stability, security, and flexibility. It has been a foundational system in the development of many modern operating systems.

**b. How to view your username in Unix? Explain your answer.**

Ans:- To view your username in Unix, you can use the `**whoami**` command. This command displays the username associated with the currently logged-in user. It works by querying the system for the user's identity and then prints it to the terminal. For example, if you run `whoami`, it will show your username.

**c. What is a File in Unix?**

Ans:-In Unix, a file is a basic unit of storage that can contain data, text, programs, or any other type of information. Files are organized within directories, forming a hierarchical file system structure. Each file is identified by a unique name within its directory, and it may have various attributes, such as permissions, ownership, and timestamps. Files can be manipulated, read, written, and executed by various Unix commands and programs.

**d. List any TWO (2) basic UNIX commands for working with files.**

Ans:- Two basic UNIX commands for working with files are:-

**1. `ls`:** The `ls` command is used to list the files and directories in the current directory. It provides information such as file names, permissions, sizes, and timestamps. For example, `ls -l` displays a detailed listing.

**2. `cp`:** The `cp` command is used to copy files and directories from one location to another. You specify the source file/directory and the destination where you want to create a copy. For example, `cp file.txt newfile.txt` copies `file.txt` to `newfile.txt`.

**e. list any FIVE (5) basic UNIX users commands.**

Ans:- Five basic UNIX user commands are:

**1. `pwd`:** The `pwd` command prints the current working directory to the terminal, showing you the full path to your current location in the file system.

**2. `cd`:** The `cd` command is used to change the current directory. You provide the name of the directory you want to navigate to as an argument. For example, `cd /home/user` changes to the `/home/user` directory.

**3. `mkdir`:** The `mkdir` command is used to create a new directory. You specify the directory name as an argument. For example, `mkdir new_directory` creates a directory named "new_directory."

**4. `rm`:** The `rm` command is used to remove files and directories. Be cautious when using it, as it deletes files permanently. For example, `rm file.txt` removes the file "file.txt."

**5. `man`:** The `man` command is used to access the manual pages for other commands. You can type `man` followed by the command name to get detailed information on how to use that command. For example, `man ls` displays the manual page for the `ls` command.

**f. Differentiate between multiuser and multitasking in UNIX operating system.**

| Multiuser | Multitasking |
|---|---|
| Multiuser means multiple users can use the system concurrently, each with their own user account and resources. | Multitasking means that a single user or multiple users can run multiple processes or programs concurrently on the same system. |
| Users are isolated from each other, with separate user accounts, home directories, and permissions. | Users or processes share system resources, but they are kept separate and can run concurrently without interfering with each other. |
| Resources such as CPU time, memory, and file access are allocated among multiple users. | Resources are allocated among multiple processes or tasks within a single user's session. |
| Designed to support multiple users working on the same system, each with their own workspaces and data. | Designed to maximize the utilization of system resources by allowing efficient switching between multiple running processes. |
| Examples include 'who' and 'w' commands to check who is logged in and 'su' to switch user accounts. | Examples include 'ps' to list processes, 'top' to monitor system activity, and 'fg' to bring a background process to the foreground. |

**Q.NO.2.**

**a. Discuss any TWO (2) uses of C programming.**

Ans:- Two uses of C programming:

**1. System Software Development:** C is commonly used for developing system software like operating systems, device drivers, and firmware due to its low-level capabilities and efficient memory management.

**2. Application Software:** C is also used for building various application software, including games, database management systems, and scientific simulations, thanks to its high performance and portability.

**b. Compute the values of the following C expressions assuming that a, b and c are integer variables as declared below:**

**int a=10, b=12, c=2;**

**i. x=a-b/3+c*2-1**

**ii. y=a-b/(3+c)*(2-1)**

**iii. z = a - (b / (3 + c) *2) - 1**

Ans:- Computing the values of the C expressions:

i. x = a - b/3 + c*2 - 1

x = 10 - 12/3 + 2*2 - 1

x = 10 - 4 + 4 - 1

x = 9

ii. y = a - b/(3+c)*(2-1)

y = 10 - 12/(3+2)*(2-1)

y = 10 - 12/5*(1)

y = 10 - 2*1

y = 8

iii. z = a - (b / (3 + c) * 2) - 1

z = 10 - (12 / (3 + 2) * 2) - 1

z = 10 - (12 / 5 * 2) - 1

z = 10 - (24 / 5) - 1

z = 10 - 4.8 - 1

z = 4.2 (Note: The result is a floating-point value as division involves floating-point arithmetic.)

**c. List FIVE (5) rules of naming identifiers.**

Ans:- Five rules of naming identifiers in C programming:

1. Identifiers must begin with a letter (uppercase or lowercase) or an underscore.

2. Following the initial character, identifiers can contain letters, digits, and underscores.

3. Identifiers are case-sensitive, meaning "myVariable" and "myvariable" are treated as distinct identifiers.

4. C language keywords (e.g., int, if, else) cannot be used as identifiers.

5. Identifiers should be meaningful and relevant to the purpose of the variable or function they represent.

**d. Write down a C expression corresponding to each of the following mathematical expressions:**

**i. —c+(c^2-4ac) /2a**

**ii. t/k — tk + 2/3s^2**

Ans:- C expressions corresponding to the given mathematical expressions:

i. -c + (c*c - 4*a*c) / (2*a)

Expression in C: `-c + (c*c - 4*a*c) / (2*a)`

ii. (t/k) - (t*k) + (2/(3*s*s))

Expression in C: `(t/k) - (t*k) + (2/(3*s*s))`

**e. List any FOUR (4) examples of identifiers in C programming.**

Ans:-Four examples of identifiers in C programming:

1. variableName
2. MAX_VALUE
3. _count
4. calculateArea

**Q.NO. 3.**

**a. What will be the output of following program?**

```
#include <stdio.h>
enum birds (SPARROW, PEACOCK, PARROT};
enum animals (TIGER = 8, LION, RABBIT, ZEBRA};
int main()
{
enum birds m = TIGER;
int k;
k = m;
printf("%d\n", k);
return 0;
}
```

Ans:- 8

**b. What is the difference between the following two C programming codes? Explain your answer.**

```
1. #include <stdio.h> //Program 1
2.     int main()
3.     {
4.         int d, a 1, b = 2;
5.         d = a++ +++b;
6.         printf("%d %d %d", d, a, b);
7.     }
```

**Program 1**

```
1. #include <stdio.h> //Program 2
2.     int main()
3.     {
4.         int d, a = 1, b = 2;
5.         d = a++ +++b;
6.         printf("%d %d %d", d, a, b);
7.     }
```

**Program 2**

Ans:-

Program 1 has a typo in the initialization of `a`. It should be `int a = 1`, but there is a space between `a` and `1`, which makes it `int a 1`. This will result in a compilation error.

Program 2 initializes `a` correctly as `int a = 1`.

**C. What is the output of the following program?**

```
//Example 1
#include <stdio.h>
int main()
{
int* pc, c;

c = 22;
printf("Address of c: %p\n", &c);
printf("Value of c: %d\n\n", c); // 22

pc = &c;
printf("Address of pointer pc: %p\n", pc);
printf("Content of pointer pc: %d\n\n", *pc); // 22

C = 11;
printf("Address of pointer pc: %p\n", pc);
printf("Content of pointer pc: %d\n\n", *pc); // 11

*pc = 2;
printf("Address of c: %p\n", &c);
printf("Value of c: %d\n\n", c); // 2
return 0;
}
```

Ans:-

Address of c: c

Value of c: 22

Address of pointer pc: c

Content of pointer pc: 22

Address of pointer pc: c

Content of pointer pc: 11

Address of c: c

Value of c: 2

**d. Identify the errors and their types in the following C Programming codes:**

**i. #include<stdio.h>**

```
main() {
        printf("Hello World")
}
```

Ans:-

Error: Missing semicolon at the end of the `printf` statement.

Error Type: Syntax error.

**ii. #include<stdio.h>**

```
main() {
        int x = 52;
        int y = 0;
        printf("Div: %f", x/y);
        }
```

Ans:-

Error: Division by zero is not allowed in C, and you're trying to divide `x` by `y`, where `y` is 0.

Error Type: Runtime error.

**iii. #include<stdio.h>**

```
Main() {
        int x = 52;
        int y = 0;
        printf("Div: %f", x/y);
        }
```

Ans:-

Error: The function `Main` should be written as `main` (lowercase 'm').

Error Type: Syntax error.

**iv. #include<stdio.h>**

```
main() {
        int i;
        for(i=0; i<5; i++); {
                printf("Hello World");
        }
}
```

Ans:-

Error: The semicolon after the `for` loop terminates the loop prematurely, and the code block within braces is not part of the loop.

Error Type: Logic error.

**e. The code below is the incomplete C code. Rewrite the following code by using the function to read a line of string and to display the string.**

```
#include <stdio.h>

int main()

{

        char name[30];

        printf("Enter name: ");

        printf("Name: ");

        return 0;

}
```

Ans:-

```c
#include <stdio.h>

// Function to read a line of string
void readString(char *str, int maxLength) {
    fgets(str, maxLength, stdin);
}

// Function to display a string
void displayString(const char *str) {
    printf("Name: %s", str);
}

int main() {
    char name[30];
    printf("Enter name: ");
    readString(name, sizeof(name));
    printf("Name: ");
    displayString(name);
    return 0;
}

```

**Q.NO. 4.**

**a. Explain what is meant by looping/iteration statements in C programming.**

Ans:- Looping or iteration statements in C programming are used to execute a block of code repeatedly as long as a certain condition is met. They allow you to automate repetitive tasks, making your code more efficient and concise. The key idea behind loops is to define a condition, and as long as that condition remains true, the code within the loop will continue to execute.

**b. Discuss THREE (3) types of loops in C programming.**

Ans:- There are three main types of loops in C programming:

**1. For Loop:** The for loop is used when you know in advance how many times you want to repeat a block of code. It consists of three parts: initialization, condition, and increment/decrement, making it ideal for iterating over a range of values.

```
for (initialization; condition; increment/decrement) {
    // Code to be executed repeatedly
}
```

**2. While Loop:** The while loop repeatedly executes a block of code as long as a specified condition remains true. It's useful when you don't know in advance how many times the loop needs to run.

```
while (condition) {
    // Code to be executed repeatedly
}
```

**3. Do-While Loop**: Similar to the while loop, the do-while loop also executes a block of code repeatedly based on a condition. However, it differs in that it guarantees that the code block will be executed at least once before checking the condition.

```
do {
    // Code to be executed repeatedly
} while (condition);
```

**c. By using suitable loop, write a program in C to calculate the sum of first n natural numbers. (Hint: Positive integers 1,2,3... are known as natural numbers).**

```c
#include <stdio.h>

int main() {
    int n, sum = 0;

    printf("Enter a positive integer n: ");
    scanf("%d", &n);

    // Check if n is a non-negative integer
    if (n < 0) {
        printf("Please enter a non-negative integer.\n");
    } else {
        for (int i = 1; i <= n; i++) {
            sum += i; // Add each natural number to the sum
        }

        printf("The sum of the first %d \nNatural numbers is: %d\n", n, sum);
    }

    return 0;
}
```

**d. Write a program in C to display such a pattern for n number of rows using a number which will start with the number 1 and the first and a last number of each row will be 1. The pattern is as follows:**

**1**

**121**

**12321**

```c
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of rows: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n - i; j++) {
            printf(" ");
        }

        for (int j = 1; j <= i; j++) {
            printf("%d", j);
        }

        for (int j = i - 1; j >= 1; j--) {
            printf("%d", j);
        }

        printf("\n");
    }

    return 0;
}
```

**e. List any FOUR (4) major operations you can perform on files.**

Ans:- Four major operations you can perform on files in C programming are:

**1. Opening a File**: To work with a file, you need to open it. You can open a file for reading, writing, or both. This operation establishes a connection between your program and the file on the storage medium.

**2. Reading from a File:** Once a file is open for reading, you can read data from it. This operation allows you to retrieve data from the file and use it in your program.

**3. Writing to a File:** When a file is open for writing, you can write data to it. This operation lets you save data generated by your program to a file on the storage medium.

**4. Closing a File:** After performing read or write operations on a file, it's essential to close the file. This ensures that any changes are saved, and resources are released.

**f. Write a syntax for opening a file in standard input and output (1/0).**

Ans:- In C programming, to open a file for standard input (0) and standard output (1), you can use the following syntax in one line:

**freopen("filename", "r", stdin); freopen("filename", "w", stdout);**