Q.NO. 1.

a. The following questions refer to the screen illustrated below that shows the partial result of Is-al command. State appropriate UNIX command(s) to answer the following questions.

```
-rwxr-xr-x 1  alexa student 23068 Jan 26 19:98 archive.sh
-rw-rw-r— 1 alexa student 12878 Feb 24 21:58 personal.txt
-rw-rw-r— 1 alexa student 2645 Jun 30 08:48 personnel.txt
-rw-r—r— 1 alexa student 168 Apr 17 11:51 university.html
drwxr-xr-x 2 alexa student 1024 Mar 18 16:27 assign
-rwxr-xr-x 1 alexa student 2645 Aug 4 11:03 program
```

Quotes (" -) were added to highlight names and can be excluded.

i. Search for the expression "grade" in "university.html".

Ans:- grep "grade" university.html

ii. Give full permission to the 'personnel.txt"

Ans:- chmod 777 personnel.txt

iii. Remove all permissions of the "personal.txt"

Ans:- chmod 000 personal.txt

iv. Run "program" executable file and take input from "customer" file.

Ans:- ./program < customer

b. Write a C program that asks the user to enter a file name and delete a filename using Linux command according to the input file name.

Ans:-

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

int main() {
    char filename[100];

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

    if (remove(filename) == 0) {
        printf("File '%s' deleted successfully.\n", filename);
    } else {
        printf("Unable to delete the file '%s'.\n", filename);
    }

    return 0;
}
```

Q.NO. 2.

a. Define the following terms:

i. Strings **ii. Call by value**

iii. Variables **iv. Datatypes**

v. Function definition

Ans:-

i. Strings:- Strings are sequences of characters, typically used to represent text in programming. They can consist of letters, numbers, symbols, or any combination thereof. In many programming languages, strings are enclosed in quotation marks (e.g., "Hello, World!").

ii. Call by Value:- Call by value is a method of passing arguments to a function in programming. When you pass a value to a function using call by value, a copy of the value is created, and the function operates on this copy. Any changes made to the parameter within the function do not affect the original value outside of the function. This method is commonly used in languages like C and Java.

iii. Variables:- Variables are symbols or identifiers used to store data values in a program. They provide a way to name and reference data in memory. Variables can hold various types of data, such as numbers, text, and more complex data structures. They are a fundamental concept in programming and are used to store and manipulate data during program execution.

iv. Datatypes:- Datatypes are a classification system used in programming to categorize and specify the type of data that a variable can hold or a function can return. Different programming languages support various datatypes, including integers, floating-point numbers, strings, booleans, arrays, and more. Datatypes determine the kind of operations that can be performed on the data and how it is stored in memory.

v. Function Definition:- A function definition is a part of a program where a specific set of instructions is written to define how a function should behave. It includes the function's name, its parameters (if any), and the code that gets executed when the function is called. Function definitions are used to encapsulate a block of code so that it can be reused multiple times in a program, promoting code modularity and reusability. Functions are a fundamental building block of structured programming.

b. Write a C program to find how many consonants and vowels in the string below.

char name []="GOOD MORNING";

Ans:-

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

int main() {
    char name[] = "GOOD MORNING";
    int vowels = 0;
    int consonants = 0;

    for (int i = 0; name[i] != '\0'; i++) {
        name[i] = toupper(name[i]);
    }

    for (int i = 0; name[i] != '\0'; i++) {
        if (name[i] >= 'A' && name[i] <= 'Z') {
            if (name[i] == 'A' || name[i] == 'E' || name[i] == 'I' || name[i] == 'O' || name[i] == 'U') {
                vowels++;
            } else {
                consonants++;
            }
        }
    }

    printf("Number of vowels: %d\n", vowels);
    printf("Number of consonants: %d\n", consonants);

    return 0;
}

```

Q.NO. 3.

a. What is the output of the following C Programs?

i. Suppose the input values are 10, 10, 20, 30, 30.

```
#include<stdio.h>
int main(void)
{
        int arr(5),  i;
        printf ("\n Enter five values: ");
        for (i=0; i<5; i++)
                scanf ("%d", aarr(i));
        printf ("\n The output is:");
        for (1=4; i>=0;
                printf ("%4d", *(arr+i));
}
```

Ans:-

Enter five values: 10 10 20 30 30

The output is: 30 30 20 10 10

ii. Suppose the input is 54.

```
#include<stdio.h>
void cbr(int *np);
int main(void)
{
int num;
        printf("Please Enter a number\t");
```

```
        scanf("%d", &num);
        printf("The original value of number is %d", num44);
        cbr(&++num);
        printf("\nThe new value of number is:\t%d\n", ++num);
         return 0;
}
void cbr(int *np)
{
        printf("\nThe value in function is: \Mt", *np);
        *np = *np * (*np) * (*np);
}
```

Ans:-

Please Enter a number 54

The original value of number is 2376

The value in function is: 55

The new value of number is: 56

b. Explain THREE (3) types of programming errors

Ans:- Here are three types of programming errors explained briefly:

1. Syntax Errors: Syntax errors are the most common type of programming errors. They occur when the code violates the rules of the programming language's syntax. This could be a missing semicolon, a typo, or incorrect indentation. These errors prevent the code from running at all.

2. Runtime Errors: Runtime errors occur when the code is syntactically correct but encounters an issue while it's running. These can include division by zero, trying to access an item in an array that doesn't exist, or using a variable before it's been initialized. Runtime errors can cause the program to crash or produce unexpected results.

3. Logical Errors: Logical errors are the trickiest to spot. They occur when the code runs without any syntax or runtime errors, but it doesn't produce the expected output. These errors are often the result of flawed logic or incorrect algorithms in the code. Debugging logical errors can be challenging because the code appears to be correct, but it doesn't perform as intended.

c. Write a program using nested for loop to produce the following output:

10 9 8 7 6

9 8 7 6 5

8 7 6 5 4

7 6 5 4 3

6 5 4 3 2

Ans:-

```c
#include <stdio.h>

int main() {
    int rows = 5;
    int cols = 5;
    int num = 10;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", num);
            num--;
        }
        num += 5;
        printf("\n");
    }

    return 0;
}

```

4. Write a C program that reads some integer numbers from the input and saves it in an array. Then, the program calls a function that finds the minimum number value and returns it to the main program. Finally, the program prints the minimum number value. Assume that the maximum number of integer numbers is 10.

Example of output;

Type list[0]: 20

Type list[1]: 10

Type list[2]: 4

Type list[3]: 30

Type list[4]: 21

Minimum is: 4

Ans:-

```c
#include <stdio.h>

// Function to find the minimum value in an array
int findMinimum(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    return min;
}
```

```c
int main() {
    int list[10];
    int n;

    // Prompt the user to input integer numbers
    printf("Enter up to 10 integer numbers (enter -1 to stop):\n");

    // Read the numbers and store them in the array
    int i = 0;
    while (i < 10) {
        printf("Type list[%d]: ", i);
        scanf("%d", &n);

        if (n == -1) {
            break; // Stop input when -1 is entered
        }

        list[i] = n;
        i++;
    }

    if (i == 0) {
        printf("No numbers entered.\n");
    } else {
        // Call the findMinimum function to find the minimum value
        int minimum = findMinimum(list, i);
        printf("Minimum is: %d\n", minimum);
    }

    return 0;
}
```