Question 1:

Write a program that swaps the values of two variables using pointers. Your program should take two integer inputs from the user and display the swapped values as output.

Input Format:

- The program will prompt the user to enter the first number.
- The user should input an integer as the first number.
- The program will then prompt the user to enter the second number.
- The user should input an integer as the second number.

Output Format:

- The program will display the original values of the two numbers before swapping.

Before swapping:

- First number: <num1>
- Second number: <num2>
- Here, <num1> and <num2> will be replaced with the actual values entered by the user.
- After swapping the values of the two numbers, the program will display the swapped values.

After swapping:

- First number: <new_num1>
- Second number: <new_num2>

Title for Question 1: Swap Values using Pointers

Solution:

```
#include <iostream>

using namespace std; // Add this line

void swapUsingPointers(int* ptr1, int* ptr2) {
    int temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

int main() {
    int num1, num2;

    //cout << "Enter the first number: ";
    cin >> num1;

    //cout << "Enter the second number: ";
    cin >> num2;

    cout << "Before swapping:" << endl;
```

```
    cout << "First number: " << num1 << endl;
    cout << "Second number: " << num2 << endl;

    // Call the swap function with pointers
    swapUsingPointers(&num1, &num2);

    cout << "After swapping:" << endl;
    cout << "First number: " << num1 << endl;
    cout << "Second number: " << num2 << endl;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	6 8	Before swapping: First number: 6 Second number: 8 After swapping: First number: 8 Second number: 6
2	10 -7	Before swapping: First number: 10 Second number: -7 After swapping: First number: -7 Second number: 10
3	10 20	Before swapping: First number: 10 Second number: 20 After swapping: First number: 20 Second number: 10
4	120 456	Before swapping: First number: 120 Second number: 456 After swapping: First number: 456 Second number: 120
5	-67 89	Before swapping: First number: -67 Second number: 89 After swapping: First number: 89 Second number: -67
6	244244 89490	Before swapping: First number: 244244 Second number: 89490 After swapping: First number: 89490 Second number: 244244

White List:

Black List:

Question 2:

Write a program that reverses a string using pointers instead of the standard string reversal functions. Your program should take a string input from the user and display the reversed string as output.

Input Format:

- The program expects the user to directly enter a string without a prompt.
- The user should enter the string. The program assumes a maximum input length of 100 characters.

Output Format:

- The program reverses the input string using pointers.
- The reversed string is displayed with the following message:
- Reversed string: <reversed_string>
- Here, <reversed_string> will be replaced with the reversed version of the input string.

Title for Question 2: Reverse a String

Solution:

```
#include <iostream>
using namespace std;
int main() {

    char inputString[100];
    cin.getline(inputString, 100);
    int length = 0;
    while (inputString[length] != '\0') {
        length++;
    }
    char* start = inputString;
     char* end = inputString + length - 1;
        while (start < end) {
        char temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
    cout << "Reversed string: " << inputString << endl;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	Amypo Technologies	Reversed string: seigolonhceT opymA
2	String implementation	Reversed string: noitatnemelpmi gnirtS
3	Hello World!	Reversed string: !dlroW olleH
4	Type casting and operator	Reversed string: rotarepo dna gnitsac epyT
5	C and C++ Programming	Reversed string: gnimmargorP ++C dna C
6	123456	Reversed string: 654321

White List:

Black List:

Question 3:

Create an array of pointers to strings and sort them in alphabetical order.

Input Format:

- The program prompts the user to enter the number of strings (n), which should be a non-negative integer.
- The user is then prompted to enter n strings, one at a time, pressing the Enter key after each string.

Output Format:

- After processing the input, the program sorts the entered strings in alphabetical order.
- It then displays the sorted strings one by one, with each string on a new line.

Title for Question 3: Array of Pointers

Solution:

```
#include <iostream>
#include <string>
#include <algorithm>

using namespace std;

int main() {
    int n;
    cin >> n;
    string** stringArray = new string*[n];
    cin.ignore();
     for (int i = 0; i < n; i++) {
        string input;
        getline(cin, input);
        stringArray[i] = new string(input);
    }
    sort(stringArray, stringArray + n, [](const string* a, const string*
        return *a < *b;
    });
    for (int i = 0; i < n; i++) {
        cout << *stringArray[i] << endl;
    }
    for (int i = 0; i < n; i++) {
        delete stringArray[i];
    }
    delete[] stringArray;
return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	3 Banana Apple Cherry	Apple Banana Cherry
2	4 Elephant Dog Cat Ant	Ant Cat Dog Elephant
3	2 Zebra Aardvark	Aardvark Zebra
4	5 Orange Lemon Grapes Apple Banana	Apple Banana Grapes Lemon Orange
5	1 Pineapple	Pineapple
6	2 Apple Banana	Apple Banana

White List:

Black List:

Question 4:

Write a Program for function that accepts a pointer to an integer as an argument and increments the value it points to by a specified amount with dynamic input from the user?

Input Format:

- The program directly reads the following input values from the user without displaying any prompts:
- An integer (number) representing the initial value.
- Another integer (incrementAmount) representing the amount by which to increment the value.

Output Format:

- The program calculates and displays the updated value of the integer pointed to by the pointer (*ptr) after incrementing it by the specified amount.

Title for Question 4: Increment Integer Value Using a Pointer with User-Defined Increment

Solution:

```
#include <iostream>
using namespace std;
void incrementValue(int* ptr, int amount) {
    if (ptr != nullptr) {
        *ptr += amount;
    }
}

int main() {
    int number;
```

```
    int incrementAmount;
    cin >> number;
    cin >> incrementAmount;
    int* ptr = &number;
    incrementValue(ptr, incrementAmount);
    cout << "Updated value: " << *ptr << endl;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	123 -123	Updated value: 0
2	42 42	Updated value: 84
3	100 -25	Updated value: 75
4	0 0	Updated value: 0
5	-5 7	Updated value: 2
6	10 5	Updated value: 15

White List:

Black List:

Question 5:

Write a program that accepts an integer from the user, stores it in a dynamically allocated integer variable, and then passes this variable to a function by value. Inside the function, modify the integer value, and print both the original and modified values. Explain the result.

Input Format:

- The user is prompted to enter two integers separated by spaces, representing n and num.

Output Format:

- The program displays the original value of the dynamically allocated integer.
- It then modifies the value num times and displays the modified value.
- Finally, it deallocates the dynamically allocated memory.

Title for Question 5: Pointer by Value

Solution:

```
#include <iostream>
using namespace std;
void modifyValue(int* ptr) {
 (*ptr)++;
}
int main() {
 int n, num;
 cin >> n;
 cin >> num;
 int* ptr = new int(n);
    cout << "Original value: " << *ptr << std::endl;
     for (int i = 0; i < num; i++) {
    modifyValue(ptr);
 }
    cout << "Modified value: " << *ptr << std::endl;
    delete ptr;
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	10 5	Original value: 10 Modified value: 15
2	10 0	Original value: 10 Modified value: 10
3	100 10	Original value: 100 Modified value: 110
4	-3 1	Original value: -3 Modified value: -2
5	100 45	Original value: 100 Modified value: 145
6	0 3	Original value: 0 Modified value: 3

White List:

Black List:

Question 6:

Create a program that allows the user to input a string of characters. Dynamically allocate memory for this string using pointers. Then, write a function that calculates and returns the length of the string. Finally, deallocate the memory for the string. Ensure that there are no memory leaks in the program?

Input Format:

- The program should read the following input from the user:
- An integer rows representing the number of rows in the matrix.
- An integer cols representing the number of columns in the matrix.
- The elements of the matrix, one element per line. There will be rows * cols elements to input.

Output Format:

- The program should display the following output:

- The transposed matrix.

Title for Question 6: Transposing a 2D Matrix using Pointers

Solution:

```
#include <iostream>
using namespace std;
// Function to transpose a 2D array using pointer-to-pointer relationship
void transpose(int** arr, int rows, int cols) {
    int** result = new int*[cols]; // Create a new array for the transpos

    for (int i = 0; i < cols; ++i) {
        result[i] = new int[rows];
        for (int j = 0; j < rows; ++j) {
            result[i][j] = arr[j][i]; // Transpose the elements
        }
    }

    // Print the transposed matrix
    std::cout << "Transposed Matrix:" << std::endl;
    for (int i = 0; i < cols; ++i) {
        for (int j = 0; j < rows; ++j) {
            std::cout << result[i][j] << " ";
        }
        std::cout << std::endl;
    }

    // Deallocate memory for the transposed matrix
    for (int i = 0; i < cols; ++i) {
        delete[] result[i];
    }
    delete[] result;
}

int main() {
    int rows, cols;
    //std::cout << "Enter the number of rows: ";
    std::cin >> rows;
    //std::cout << "Enter the number of columns: ";
    std::cin >> cols;

    int** arr = new int*[rows]; // Create a 2D array

    // Input matrix elements from the user
    //std::cout << "Enter the elements of the matrix:" << std::endl;
    for (int i = 0; i < rows; ++i) {
        arr[i] = new int[cols];
        for (int j = 0; j < cols; ++j) {
            std::cin >> arr[i][j];
        }
    }

    /*// Print the original matrix
    std::cout << "Original Matrix:" << std::endl;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
```

```
            std::cout << arr[i][j] << " ";
        }
        std::cout << std::endl;
    */

    // Transpose the matrix using pointer manipulations
    transpose(arr, rows, cols);

    // Deallocate memory for the original matrix
    for (int i = 0; i < rows; ++i) {
        delete[] arr[i];
    }
    delete[] arr;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	3 3 6 7 8 4 5 6 1 2 6	Transposed Matrix: 6 4 1 7 5 2 8 6 6
2	4 4 89 54 32 23 67 89 54 123 56 098 453 123 67 89 54 32 125	Transposed Matrix: 89 67 56 67 54 89 98 89 32 54 453 54 23 123 123 32
3	3 2 5 6 8 9 1 2	Transposed Matrix: 5 8 1 6 9 2
4	2 2 78 90 78 43	Transposed Matrix: 78 78 90 43
5	1 1 45	Transposed Matrix: 45
6	2 3 6 7 8 3 4 1	Transposed Matrix: 6 3 7 4 8 1

White List:

Black List:
