LAB TASK(week-3)

Name: Abdul Moiz

Sap I'd: 54482

Question #1

```
#include <iostream>
using namespace std;

int main() {
    int rows, cols;

    // Input the dimensions of the 2D array
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> cols;

    // Dynamically allocate memory for the 2D array
    int** array = new int*[rows];
    for (int i = 0; i < rows; ++i) {
        array[i] = new int[cols];
    }
```

```
// Input data for the 2D array
cout << "Enter the elements of the " << rows << "x" << cols << " array:" << endl;
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        cout << "Element [" << i << "][" << j << "]: ";
        cin >> array[i][j];
    }
}

// Calculate sum, product, and average
int sum = 0;
long long product = 1; // Use long long to handle large products
int totalElements = rows * cols;

for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        sum += array[i][j];
        product *= array[i][j];
    }
}

double average = static_cast<double>(sum) / totalElements;

// Display results
```

```
    cout << "Sum of all elements: " << sum << endl;
    cout << "Product of all elements: " << product << endl;
    cout << "Average of all elements: " << average << endl;

    // Free dynamically allocated memory
    for (int i = 0; i < rows; ++i) {
        delete[] array[i];
    }
    delete[] array;

    return 0;
}
```

Question #2

```
#include <iostream>
using namespace std;

// Function to swap values using pointers
void swap(int* a, int* b) {
    int temp = *a; // Store the value pointed to by a in temp
    *a = *b;      // Assign the value pointed to by b to the location pointed to by a
    *b = temp;    // Assign the value stored in temp to the location pointed to by b
}
```

```
int main() {
    int x, y;

    // Input values for x and y
    cout << "Enter the value of x: ";
    cin >> x;
    cout << "Enter the value of y: ";
    cin >> y;

    // Display values before swapping
    cout << "Before swapping: x = " << x << ", y = " << y << endl;

    // Call the swap function
    swap(&x, &y);

    // Display values after swapping
    cout << "After swapping: x = " << x << ", y = " << y << endl;

    return 0;
}
```

Question #3:

```
#include <iostream>
using namespace std;

int main() {
    const int SIZE = 10;
    int values[SIZE];

    // Input values into the array
    cout << "Enter " << SIZE << " values:" << endl;
    for (int i = 0; i < SIZE; ++i) {
        cout << "Value " << (i + 1) << ": ";
        cin >> values[i];
    }

    // Initialize min and max with the first element
    int min = values[0];
    int max = values[0];

    // Find the smallest and largest values
    for (int i = 1; i < SIZE; ++i) {
        if (values[i] < min) {
            min = values[i];
        }
        if (values[i] > max) {
            max = values[i];
        }
    }

    // Display the results
    cout << "The smallest value is: " << min << endl;
    cout << "The largest value is: " << max << endl;

    return 0;
}
```

Question #4

```
#include <iostream>
#include <iomanip> // For std::fixed and std::setprecision
using namespace std;

int main() {
    const int MONTHS = 12;
    double rainfall[MONTHS];

    // Input rainfall data
    cout << "Enter the total rainfall for each of the 12 months:" << endl;
    }

    for (int i = 0; i < MONTHS; ++i) {
        cout << "Month " << (i + 1) << ": ";
        cin >> rainfall[i];
    }

    // Calculate total rainfall, average monthly rainfall, and find the month with
highest and lowest rainfall
    double totalRainfall = 0.0;
    double highestRainfall = rainfall[0];
    double lowestRainfall = rainfall[0];
    int highestMonth = 0;
    int lowestMonth = 0;

    for (int i = 0; i < MONTHS; ++i) {
        totalRainfall += rainfall[i];
        if (rainfall[i] > highestRainfall) {
            highestRainfall = rainfall[i];
            highestMonth = i;
        }
        if (rainfall[i] < lowestRainfall) {
            lowestRainfall = rainfall[i];
            lowestMonth = i;
        }
    }
    double averageRainfall = totalRainfall / MONTHS;


    // Display results
    cout << fixed << setprecision(2); // Set precision for floating-point output
    cout << "Total rainfall for the year: " << totalRainfall << " inches" << endl;
    cout << "Average monthly rainfall: " << averageRainfall << " inches" << endl;
    cout << "Month with highest rainfall: Month " << (highestMonth + 1) << " with "
<< highestRainfall << " inches" << endl;
    cout << "Month with lowest rainfall: Month " << (lowestMonth + 1) << " with "
<< lowestRainfall << " inches" << endl;


    return 0;
}
```

Question #5

```
#include <iostream>
using namespace std;


const int ROWS = 3;
const int COLS = 4;


// Function to get the total of all elements in the 2D array
int getTotal(int array[ROWS][COLS]) {
    int total = 0;
    for (int i = 0; i < ROWS; ++i) {
        for (int j = 0; j < COLS; ++j) {
            total += array[i][j];
        }
    }
    return total;
}

// Function to calculate the average of all values in the 2D array
double getAverage(int array[ROWS][COLS]) {
    int total = getTotal(array);
    return static_cast<double>(total) / (ROWS * COLS);
}

// Function to get the total of a specified row
int getRowTotal(int array[ROWS][COLS], int row) {
    int total = 0;
    for (int j = 0; j < COLS; ++j) {
        total += array[row][j];
    }
    return total;
}

// Function to get the total of a specified column
```

```
int getColumnTotal(int array[ROWS][COLS], int col) {
    int total = 0;
    for (int i = 0; i < ROWS; ++i) {
        total += array[i][col];
    }
    return total;
}

// Function to get the highest value in a specified row
int getHighestInRow(int array[ROWS][COLS], int row) {
    int highest = array[row][0];
    for (int j = 1; j < COLS; ++j) {
        if (array[row][j] > highest) {
            highest = array[row][j];
        }
    }
    return highest;
}

// Function to get the highest value in a specified column
int getHighestInColumn(int array[ROWS][COLS], int col) {
    int highest = array[0][col];
    for (int i = 1; i < ROWS; ++i) {
        if (array[i][col] > highest) {
            highest = array[i][col];
        }
    }
    return highest;
}

int main() {
    // Initialize a 2D array with test data
    int array[ROWS][COLS] = {
        {10, 20, 30, 40},
        {50, 60, 70, 80},
        {90, 100, 110, 120}
    };

    // Perform operations
    int row = 1; // Specify row index for operations
    int col = 2; // Specify column index for operations

    cout << "Total of all elements: " << getTotal(array) << endl;
    cout << "Average of all elements: " << getAverage(array) << endl;
    cout << "Total of row " << row << ": " << getRowTotal(array, row) << endl;
    cout << "Total of column " << col << ": " << getColumnTotal(array, col) << endl;
    cout << "Highest value in row " << row << ": " << getHighestInRow(array, row) <<
endl;
```

```
    cout << "Highest value in column " << col << ": " << getHighestInColumn(array,
col) << endl;


    return 0;
}
```

Question #6

```
#include <iostream>
using namespace std;


int main() {
    int size;


    // Input the size of the array
    cout << "Enter the number of elements: ";
    cin >> size;


    // Dynamically allocate memory for the array
    int* array = new int[size];


    // Input values for the array
    cout << "Enter " << size << " integers:" << endl;
    for (int i = 0; i < size; ++i) {
        cout << "Element " << (i + 1) << ": ";
        cin >> array[i];
    }

    // Calculate the sum of odd integers
    int sumOfOdd = 0;
    for (int i = 0; i < size; ++i) {
        if (array[i] % 2 != 0) { // Check if the number is odd
            sumOfOdd += array[i];
        }
    }

    // Display the result
    cout << "Sum of odd integers: " << sumOfOdd << endl;

    // Free dynamically allocated memory
    delete[] array;

    return 0;
}
```

Question #7

```
#include <iostream>
```

```
using namespace std;

int main() {
    int value = 42;      // Define an integer variable
    int* ptr = &value;   // Define a pointer variable and assign the address of 'value'
to it

    // Access and display the value using the pointer
    cout << "Value of the variable: " << value << endl;
    cout << "Address of the variable: " << ptr << endl;
    cout << "Value accessed through pointer: " << *ptr << endl;

    return 0;
}
```

Question #8

```
#include <iostream>
using namespace std;

int main() {
    int a, b;        // Declare two integer variables
    int* ptrA = nullptr; // Declare pointers and initialize to nullptr
    int* ptrB = nullptr;
    // Input values for a and b
    cout << "Enter value for a: ";
    cin >> a;
    cout << "Enter value for b: ";
    cin >> b;

    // Assign addresses of a and b to pointers
    ptrA = &a;
    ptrB = &b;

    // Display values using pointers
    cout << "Value of a using pointer ptrA: " << *ptrA << endl;
    cout << "Value of b using pointer ptrB: " << *ptrB << endl;

    return 0;
}
```

Question #9

```
#include <iostream>
#include <cmath>    // For pow function
using namespace std;
```

```
// Function to display the menu and handle user choice
void Menu() {
    int choice, a, b;
    do {
        // Display menu options
        cout << "\nCalculator Menu:\n";
        cout << "1. Addition\n";
        cout << "2. Subtraction\n";
        cout << "3. Multiplication\n";
        cout << "4. Division\n";
        cout << "5. Power\n";
        cout << "6. Exit\n";
        cout << "Enter your choice (1-6): ";
        cin >> choice;

        // Handle user choice
        switch (choice) {
            case 1:
                cout << "Enter two integers: ";
                cin >> a >> b;
                cout << "Result: " << Addition(a, b) << endl;
                break;
            case 2:
```

```
    cout << "Enter two integers: ";
    cin >> a >> b;
    cout << "Result: " << Subtraction(a, b) << endl;
    break;
case 3:
    cout << "Enter two integers: ";
    cin >> a >> b;
    cout << "Result: " << Multiplication(a, b) << endl;
    break;
case 4:
    cout << "Enter two integers (denominator must not be zero): ";
    cin >> a >> b;
    if (b != 0) {
        cout << "Result: " << Division(a, b) << endl;
    } else {
        cout << "Error: Division by zero is not allowed.\n";
    }
    break;
case 5:
    cout << "Enter base and exponent: ";
    cin >> a >> b;
    cout << "Result: " << Pow(a, b) << endl;
    break;
case 6:
```

```
                cout << "Exiting the program.\n";
                break;
            default:
                cout << "Invalid choice. Please select a number between 1 and 6.\n";
                break;
        }
    } while (choice != 6); // Repeat until the user chooses to exit
}

// Function to add two integers
int Addition(int a, int b) {
    return a + b;
}

// Function to subtract the second integer from the first
int Subtraction(int a, int b) {
    return a - b;
}

// Function to multiply two integers
int Multiplication(int a, int b) {
    return a * b;
}
```

```
// Function to divide the first integer by the second
double Division(int a, int b) {
    return static_cast<double>(a) / b;
}

// Function to calculate the power of a number
int Pow(int number, int pow) {
    return static_cast<int>(std::pow(number, pow));
}

int main() {
    // Call the Menu function
    Menu();
    return 0;
}
```