```
//-----------------------------------------------------------------------
// File name: Exercise.cpp
// Assign ID:
// Due Date: 28/08/24 at 11pm
//
// Purpose: Functions and Function overloading
//
// Author: Mr. KEO Sopahnit
//-----------------------------------------------------------------------

Exercise_1
#include <iostream>
#include <algorithm>
using namespace std;

// Function to initialize the matrix
void initializeMatrix(int matrix[][100], int size, int value) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix[i][j] = value;
        }
    }
}

void initializeMatrix(double matrix[][100], int size, double value) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix[i][j] = value;
        }
    }
}
// Function to initialize of square matrix
void initializeMatrix(char matrix[][100], int size, char value) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix[i][j] = value;
        }
    }
}

// Function to output the matrix
template <typename T>
void outputMatrix(T matrix[][100], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
// Function to determine the maximum and minimum elements on the main
diagonal
template <typename T>
void findMaxMinDiagonal(T matrix[][100], int size, T &max, T &min) {
```

```
    max = matrix[0][0];
    min = matrix[0][0];
    for (int i = 1; i < size; i++) {
        if (matrix[i][i] > max) {
            max = matrix[i][i];
        }
        if (matrix[i][i] < min) {
            min = matrix[i][i];
        }
    }
}

// Function to sort each row in ascending order
template <typename T>
void sortRows(T matrix[][100], int size) {
    for (int i = 0; i < size; i++) {
        sort(matrix[i], matrix[i] + size);
    }
}
```

Exercise_2

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

// Maximum value in a 1D array
int findMax(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
string findMax(string arr[], int size) {
    string max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Maximum value in a 2D array
int findMax(int arr[][100], int rows, int cols) {
    int max = arr[0][0];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] > max) {
                max = arr[i][j];
            }
        }
    }
    return max;
}
string findMax(string arr[][100], int rows, int cols) {
    string max = arr[0][0];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] > max) {
                max = arr[i][j];
            }
        }
    }
    return max;
}

// Maximum value in a 3D array
int findMax(int arr[][100][100], int depth, int rows, int cols) {
    int max = arr[0][0][0];
    for (int i = 0; i < depth; i++) {
        for (int j = 0; j < rows; j++) {
```

```
            for (int k = 0; k < cols; k++) {
                if (arr[i][j][k] > max) {
                    max = arr[i][j][k];
                }
            }
        }
    }
    return max;
}

// Maximum value of two integers
int findMax(int a, int b) {
    return (a > b) ? a : b;
}

// Maximum value of three integers
int findMax(int a, int b, int c) {
    int max = (a > b) ? a : b;
    return (max > c) ? max : c;
}
//Exercise_2
// Template function to find the maximum value in a 1D array
template <typename T>
T findMax(T arr[], int size) {
    T max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Template function to find the maximum value in a 2D array
template <typename T>
T findMax(T arr[][100], int rows, int cols) {
    T max = arr[0][0];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (arr[i][j] > max) {
                max = arr[i][j];
            }
        }
    }
    return max;
}

// Template function to find the maximum value in a 3D array
template <typename T>
T findMax(T arr[][100][100], int depth, int rows, int cols) {
    T max = arr[0][0][0];
    for (int i = 0; i < depth; i++) {
        for (int j = 0; j < rows; j++) {
            for (int k = 0; k < cols; k++) {
```

```
                if (arr[i][j][k] > max) {
                    max = arr[i][j][k];
                }
            }
        }
    }
    return max;
}

// Template function to find the maximum of two values
template <typename T>
T findMax(T a, T b) {
    return std::max(a, b); // Using std::max for conciseness
}

// Template function to find the maximum of three values
template <typename T>
T findMax(T a, T b, T c) {
    return max(a,max(b, c));
}
```

```
Exercise_3
#include <iostream>
using namespace std;

// Function to initialize the matrix
template <typename T>
void initializeMatrix(T matrix[][100], int size, T value) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix[i][j] = value;
        }
    }
}
// Function to output the matrix
template <typename T>
void outputMatrix(T matrix[][100], int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}
// Function to determine the maximum and minimum elements on the main
diagonal
template <typename T>
void findMaxMinDiagonal(T matrix[][100], int size, T &max, T &min)
{
    max = matrix[0][0];
    min = matrix[0][0];
    for (int i = 1; i < size; i++)
    {
        if (matrix[i][i] > max)
        {
            max = matrix[i][i];
        }
        if (matrix[i][i] < min)
        {
            min = matrix[i][i];
        }
    }
}

// Function to sort each row in ascending order
template <typename T>
void sortRows(T matrix[][100], int size)
{
    for (int i = 0; i < size; i++)
    {
        sort(matrix[i], matrix[i] + size);
    }}
```

```
//--------------------------------------------------------------------
// File name: Exercise.cpp
// Assign ID:
// Due Date: 28/08/24 at 11pm
//
// Purpose: Sorting arrays
//
// Author: Mr. KEO Sopahnit
//--------------------------------------------------------------------

Exercise_1
//Bubble Short
void swapping_number(int &a, int &b){
    int t;
    t = a;
    a = b;
    b = t;
}
void sort_number(int arr[], int count){
    for(int i=0;i<count;i++){
        for(int j=0;j<count-i-1;j++){
            if(arr[j]<arr[j+1]){
                swapping_number(arr[j],arr[j+1]);
            }
        }
    }
}
```

Exercise_2

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

// Function to perform insertion sort on an array
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
// Function to print the array
void displayArrray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int n = 5;

    cout << "Original array: ";
    displayArrray(arr, n);

    insertionSort(arr, n);

    cout << "Sorted array: ";
    displayArrray(arr, n);

    return 0;
}
```

```
Exercise_3
#include <iostream>
using namespace std;

void outputMarks(int marks[], int size);
void retakeExam(int marks[], int size);
void checkScholarship(int marks[], int size);
void displayMenu();

enum Menu{ EXIT, OUTPUT, RETAKE_EXAM, CHECK_SCHOOLARSHIP };

int main() {
    const int size = 10;
    int marks[size];

    // Input 10 marks
    cout << "Enter 10 marks for the student:" << endl;
    for (int i = 0; i < size; i++) {
        cin >> marks[i];
    }

    int choice;
    do {
        displayMenu();
        cin >> choice;
        switch (choice) {
            case OUTPUT:
                outputMarks(marks, size);
                break;
            case RETAKE_EXAM:
                retakeExam(marks, size);
                break;
            case CHECK_SCHOOLARSHIP:
                checkScholarship(marks, size);
                break;
            case EXIT:
                cout << "Exiting the program." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 4);

    return 0;
}
// Display Menu
void displayMenu(){
    cout << "\nMenu:\n";
    cout << "1. Output marks\n";
    cout << "2. Retake exam\n";
    cout << "3. Check if there is a scholarship\n";
    cout << "0. Exit\n";
    cout << "Enter your choice: ";
}
```

```
// Function to output the marks
void outputMarks(int marks[], int size) {
    cout << "Marks: ";
    for (int i = 0; i < size; i++) {
        cout << marks[i] << " ";
    }
    cout << endl;
}

// Function to retake an exam
void retakeExam(int marks[], int size) {
    int index, newMark;
    cout << "Enter the index (0 to 9) of the mark you want to change: ";
    cin >> index;

    if (index >= 0 && index < size) {
        cout << "Enter the new mark: ";
        cin >> newMark;
        marks[index] = newMark;
        cout << "Mark updated successfully." << endl;
    } else {
        cout << "Invalid index. Please try again." << endl;
    }
}

// Function to check if the student qualifies for a scholarship
void checkScholarship(int marks[], int size) {
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += marks[i];
    }
    double average = sum / size;

    cout << "Average mark: " << average << endl;
    if (average >= 10.7) {
        cout << "The student qualifies for a scholarship." << endl;
    } else {
        cout << "The student does not qualify for a scholarship." << endl;
    }
}
```

Exercise_4

```
#include<iostream>
using namespace std;
// Function to calculate the arithmetic mean of an array
double calculateMean(int arr[], int size) {
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum / size;
}

// Function to reverse part of the array
void reverseArray(int arr[], int start, int end) {
    while (start < end) {
        swap(arr[start], arr[end]);
        start++;
        end--;
    }
}

// Function to print the array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

Exercise_4

```
#include<iostream>
using namespace std;
// Function to calculate the arithmetic mean of an array
double calculateMean(int arr[], int size) {
    double sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum / size;
}

// Function to reverse part of the array
void reverseArray(int arr[], int start, int end) {
    while (start < end) {
        swap(arr[start], arr[end]);
        start++;
        end--;
    }
}

// Function to print the array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```