	Air University - Aerospace and Aviation Campus, Kamra **Department of Computer Science**

Programming Fundamental (CS111)

Assignment # 04

[CLO-4, Taxonomy Level-C4, PLO-4]

Solution

Course: BSCS-1 **Semester: 1st (**Fall 2023)

Due Date: 07/12/2023 **Total** Marks**:** 30

Instructions

1. *Plagiarism, copy & past material will lead to the cancellation of your assignment.*
2. *Write your* Name, Reg# on the first page (title page) of your submission.
3. No late submission

1.

```
//Task#1 - Program take an integer as input and give whether the number is prime
or not.
#include <iostream>
#include <cmath>
using namespace std;

// Function to check if a number is prime
bool isPrime(int num) {
    if (num <= 1) {
        return false;
    }

    // Check for factors up to the square root of the number
    for (int i = 2; i <= sqrt(num); ++i) {
        if (num % i == 0) {
            return false;
        }
    }

    return true;
}
int main() {

    int Number;
    cout << "Enter a number: ";
    cin >> Number;

    // Call the isPrime function and display the result
    if (isPrime(Number)) {
        cout << Number << " is a prime number.\n";
    } else {
        cout << Number << " is not a prime number.\n";
    }

    return 0;
}
```

2.

```
// Function to calculate the average of elements in an array
#include <iostream>
using namespace std;

double calculateAverage(int arr[], int size) {
    if (size == 0) {
        return 0.0;
    }

    int sum = 0;
    for (int i = 0; i < size; ++i) {
        sum += arr[i];
    }

    return static_cast<double>(sum) / size;
}

int main() {
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;
    int arr[size];

    cout << "Enter " << size << " elements:\n";
    for (int i = 0; i < size; ++i) {
        cout << "Element " << i + 1 << ": ";
        cin >> arr[i];
    }
    double average = calculateAverage(arr, size);
    cout << "The average of the elements is: " << average << "\n";

    return 0;
}
```

3.

```
// Function to find the maximum element in an array
#include <iostream>
using namespace std;

int findMaxElement(int arr[], int size) {
    if (size == 0) {
        return -1;
    }

    int maxElement = arr[0];
    for (int i = 1; i < size; ++i) {
        if (arr[i] > maxElement) {
            maxElement = arr[i];
        }
    }
    return maxElement;
}
int main() {
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;
    int arr[size];

    cout << "Enter " << size << " elements:\n";
    for (int i = 0; i < size; ++i) {
        cout << "Element " << i + 1 << ": ";
        cin >> arr[i];
    }
    int maxElement = findMaxElement(arr, size);

    if (maxElement != -1) {
        cout << "The maximum element in the array is: " << maxElement << "\n";
    } else {
        cout << "Error: Array is empty.\n";
    }
    return 0;
```

4.

```
// Recursive function to calculate the power of a number
#include <iostream>
using namespace std;

int power(int base, int exponent) {
    if (exponent == 0) {
        return 1;
    }
    return base * power(base, exponent - 1);
}

int main() {
    int base, exponent;
    cout << "Enter the base: ";
    cin >> base;
    cout << "Enter the exponent: ";
    cin >> exponent;

    int result = power(base, exponent);
    cout << base << " raised to the power of " << exponent << " is: " << result
<< "\n";

    return 0;
}
```

5.

```
// Function to add two matrices
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
void addMatrices(int mat1[MAX_SIZE][MAX_SIZE], int mat2[MAX_SIZE][MAX_SIZE], int
result[MAX_SIZE][MAX_SIZE], int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

void multiplyMatrices(int mat1[MAX_SIZE][MAX_SIZE], int mat2[MAX_SIZE][MAX_SIZE], int
result[MAX_SIZE][MAX_SIZE], int rows1, int cols1, int rows2, int cols2) {
    for (int i = 0; i < rows1; ++i) {
        for (int j = 0; j < cols2; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < cols1; ++k) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
void transposeMatrix(int mat[MAX_SIZE][MAX_SIZE], int result[MAX_SIZE][MAX_SIZE], int rows, int
cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[j][i] = mat[i][j];
        }
    }
}
```

```
void displayMatrix(int mat[MAX_SIZE][MAX_SIZE], int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int rows, cols;

    // Get user input for matrix dimensions
    cout << "Enter the number of rows for matrices: ";
    cin >> rows;
    cout << "Enter the number of columns for matrices: ";
    cin >> cols;

    // Check if the matrices are within the allowed size
    if (rows > MAX_SIZE || cols > MAX_SIZE) {
        cerr << "Error: Matrix size exceeds the maximum allowed size." << std::endl;
        return 1;
    }

    int matrix1[MAX_SIZE][MAX_SIZE];
    int matrix2[MAX_SIZE][MAX_SIZE];
    int resultMatrix[MAX_SIZE][MAX_SIZE];

    // Get user input for matrix1
    cout << "Enter elements for matrix1:" << std::endl;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cout << "Enter element at position (" << i << ", " << j << "): ";
            cin >> matrix1[i][j];
        }
    }

    // Get user input for matrix2
    cout << "Enter elements for matrix2:" << std::endl;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cout << "Enter element at position (" << i << ", " << j << "): ";
            cin >> matrix2[i][j];
        }
    }

    // Perform matrix operations
    addMatrices(matrix1, matrix2, resultMatrix, rows, cols);
    cout << "Result of matrix addition:" << std::endl;
    displayMatrix(resultMatrix, rows, cols);

    multiplyMatrices(matrix1, matrix2, resultMatrix, rows, cols, rows, cols);
    cout << "Result of matrix multiplication:" << std::endl;
    displayMatrix(resultMatrix, rows, cols);

    transposeMatrix(matrix1, resultMatrix, rows, cols);
    cout << "Transpose of matrix1:" << std::endl;
    displayMatrix(resultMatrix, cols, rows);

    return 0;
}
```