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

Programming Fundamental (CS111)

Assignment # 05

[CLO-3, Taxonomy Level-C3, PLO-3]

Solution

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

Due Date: 20/12/2023 **Total** Marks**:** 30

1.

```
//Exception handling1
#include <iostream>
using namespace std;
double add(double a, double b) {
    return a + b;
}
double subtract(double a, double b) {
    return a - b;
}
double multiply(double a, double b) {
    return a * b;
}
double divide(double a, double b) {
    if (b == 0) {
        throw ("Division by zero is not allowed.");
    }
    return a / b;
}
int main() {
    char operation;
    double operand1, operand2, result;
    cout << "Simple Calculator Program\n";
    cout << "Enter a mathematical expression (e.g., 2 + 3): ";
    try {
        cin >> operand1 >> operation >> operand2;
        if (cin.fail() || cin.eof()) {
            throw ("Invalid input. Please enter a valid expression.");
        }
        switch (operation) {
            case '+':
                result = add(operand1, operand2);
                break;
            case '-':
                result = subtract(operand1, operand2);
                break;
            case '*':
                result = multiply(operand1, operand2);
                break;
            case '/':
                result = divide(operand1, operand2);
                break;
            default:
                throw ("Invalid operation. Please enter a valid operation (+, -,
*, /).");
        }
```

2.

```
//Exception handling2
 #include <iostream>
using namespace std;
void inputMatrix(int matrix[10][10], int rows, int cols) {
   cout << "Enter the elements of the matrix:" << endl;
   for (int i = 0; i < rows; ++i) {
      for (int j = 0; j < cols; ++j) {
         cout << "Enter element at position (" << i + 1 << ", " << j + 1 << "): ";
         cin >> matrix[i][j];
      } }}
void printMatrix(int matrix[10][10], int rows, int cols) {
   cout << "Matrix:" << endl;
   for (int i = 0; i < rows; ++i) {
      for (int j = 0; j < cols; ++j) {
         cout << matrix[i][j] << " ";
      }
      cout << endl;
   }}
void addMatrices(int matrix1[10][10], int matrix2[10][10], int result[10][10], int rows, int cols) {
   for (int i = 0; i < rows; ++i) {
      for (int j = 0; j < cols; ++j) {
         result[i][j] = matrix1[i][j] + matrix2[i][j];
      }   }
}

void subtractMatrices(int matrix1[10][10], int matrix2[10][10], int result[10][10], int rows, int cols) {
   for (int i = 0; i < rows; ++i) {
      for (int j = 0; j < cols; ++j) {
         result[i][j] = matrix1[i][j] - matrix2[i][j];
      }
   }}
void multiplyMatrices(int matrix1[10][10], int matrix2[10][10], int result[10][10], 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] += matrix1[i][k] * matrix2[k][j];
         }}}}
int main() {
   int matrix1[10][10], matrix2[10][10], result[10][10];
   int rows1, cols1, rows2, cols2;
   cout << "Enter the number of rows and columns for Matrix 1:" << endl;
   cin >> rows1 >> cols1;
   inputMatrix(matrix1, rows1, cols1);

   cout << "Enter the number of rows and columns for Matrix 2:" << endl;
   cin >> rows2 >> cols2;
   inputMatrix(matrix2, rows2, cols2);

   try {
      if (rows1 != rows2 || cols1 != cols2) {
         throw "Incompatible matrix sizes for the selected operation.";
      }
      cout << "Choose operation:" << endl;
      cout << "1. Addition" << endl;
      cout << "2. Subtraction" << endl;
      cout << "3. Multiplication" << endl;

      int choice;
      cout << "Enter your choice (1/2/3): ";
      cin >> choice;
       switch (choice) {
         case 1:
            addMatrices(matrix1, matrix2, result, rows1, cols1);
            break;
         case 2:
            subtractMatrices(matrix1, matrix2, result, rows1, cols1);
```

3.

```
using namespace std;
#include <iostream>
#include <fstream>
using namespace std;
const int MAX_TEMPERATURES = 10;
double celsiusToFahrenheit(double celsius) {
    return (celsius * 9 / 5) + 32;
}

double fahrenheitToCelsius(double fahrenheit) {
    return (fahrenheit - 32) * 5 / 9;
}
int main() {
    string filename;
    cout << "Enter the name of the input file: ";
    cin >> filename;
    ifstream inputFile(filename);

    if (!inputFile.is_open()) {
        cerr << "Error: Unable to open the file." << endl;
        return 1;
    }
    double temperatures[MAX_TEMPERATURES];
    int count = 0;
    double temperature;
    try {
        while (inputFile >> temperature) {
            if (temperature < -273.15) {
                throw "Invalid temperature value: below absolute zero.";
            }

            if (count >= MAX_TEMPERATURES) {
                throw "Too many temperatures in the file. Increase
MAX_TEMPERATURES.";
            }

            temperatures[count++] = temperature;
        }
    } catch (const char* errorMessage) {
        cerr << "Error: " << errorMessage << endl;
        return 1;
    }

    inputFile.close();
    double convertedTemperatures[MAX_TEMPERATURES];
    for (int i = 0; i < count; ++i) {
        convertedTemperatures[i] = celsiusToFahrenheit(temperatures[i]);
    }
    cout << "Original Temperatures (Celsius):" << endl;
    for (int i = 0; i < count; ++i) {
        cout << temperatures[i] << " ";
    }
    cout << endl;

    cout << "Converted Temperatures (Fahrenheit):" << endl;
    for (int i = 0; i < count; ++i) {
        cout << convertedTemperatures[i] << " ";
    }
    cout << endl;
    return 0;
}
```

4.

```
#include <iostream>
#include <fstream>
using namespace std;
const int MAX_PASSWORDS = 100;
const int MIN_PASSWORD_LENGTH = 8;

bool isValidPassword(const string &password)
{
    if (password.length() < MIN_PASSWORD_LENGTH)
    {
        throw ("Invalid password: Password is too short.");
    }
    return true;
}

int main()
{
    string filename;
    cout << "Enter the name of the input file containing passwords: ";
    cin >> filename;

    ifstream inputFile(filename);

    if (!inputFile.is_open())
    {
        cerr << "Error: Unable to open the file." << endl;
        return 1;
    }
    string passwords[MAX_PASSWORDS];
    int validPasswordCount = 0;
    string password;
    try
    {
        while (inputFile >> password)
        {
            try
            {
                if (isValidPassword(password))
                {
                    if (validPasswordCount >= MAX_PASSWORDS)
                    {
                        throw ("Too many valid passwords. Increase MAX_PASSWORDS.");
                    }

                    passwords[validPasswordCount++] = password;
                }
            }
            catch (const string e)
            {
                cerr << "Invalid password: " << e << endl;
            }
        }
    }
    catch (const string e)
    {
        cerr << "Error reading passwords: " << e << endl;
        return 1;
    }
    inputFile.close();
    cout << "Summary of Valid Passwords:" << endl;
    cout << "Number of passwords read: " << (validPasswordCount + 1) << endl;
    cout << "Number of valid passwords: " << validPasswordCount << endl;

    cout << "Valid Passwords:" << endl;
    for (int i = 0; i < validPasswordCount; ++i)
  {
        cout << passwords[i] << endl;
    }
    return 0;
}
```

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <stdexcept>
using namespace std;
struct Student
{
   int id;
   string name;
   double gpa;
};
void writeStudentToFile(const Student &student, const string &filename)
{
   ofstream outFile(filename, ios::app);
   if (!outFile.is_open())
   {
      throw runtime_error("Error: Unable to open file for writing.");
   }

   outFile << student.id << " " << student.name << " " << student.gpa << endl;
   outFile.close();
}
void displayStudentDetails(const Student &student)
{
   cout << "Student ID: " << student.id << endl;
   cout << "Name: " << student.name << endl;
   cout << "GPA: " << fixed << setprecision(2) << student.gpa << endl;
}

void searchStudentByID(int searchID, const string &filename)
{
   ifstream inFile(filename);
   if (!inFile.is_open())
   {
      throw runtime_error("Error: Unable to open file for reading.");
   }

   Student student;
   bool found = false;

   while (inFile >> student.id >> student.name >> student.gpa)
   {
      if (student.id == searchID)
      {
         found = true;
         displayStudentDetails(student);
         break;
      }
   }

   if (!found)
   {
      cout << "Student with ID " << searchID << " not found." << endl;
   }

   inFile.close();
}

int main()
{
   const string filename = "student_database.txt";

   while (true)
   {
      cout << "\nStudent Database Management System\n";
      cout << "1. Add New Student\n";
      cout << "2. Display Student Details\n";
      cout << "3. Search Student by ID\n";
      cout << "4. Exit\n";
      cout << "Enter your choice (1-4): ";

      int choice;
      cin >> choice;

      switch (choice)
      {
      case 1:
```