Salman Ahmad

04072113050

BSCS 6th Sem

CS-121 OOP Assignment 3

Q1.

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

using namespace std;

class Entertainment {
private:
   string title;
   string airDate;
   string* genre;   // dynamic array for genre
   int genreCount;  // number of genres
   string type;
   int runtime;
   string country;
   string* actors;  // dynamic array for actors
   int actorCount;  // number of actors
   float rating;

public:
```

```
// Parameterized constructor
Entertainment(string _title, string _airDate, string* _genre, int _genreCount,
    string _type, int _runtime, string _country, string* _actors, int _actorCount,
    float _rating)
    : title(_title), airDate(_airDate), type(_type), runtime(_runtime),
    country(_country), rating(_rating), genreCount(_genreCount), actorCount(_actorCount) {

    // Copy genre array
    genre = new string[genreCount];
    for (int i = 0; i < genreCount; ++i) {
        genre[i] = _genre[i];
    }

    // Copy actors array
    actors = new string[actorCount];
    for (int i = 0; i < actorCount; ++i) {
        actors[i] = _actors[i];
    }
}

// Copy constructor (for genre, country, and rating only)
Entertainment(const Entertainment& other) {
    title = other.title;
    airDate = other.airDate;
    type = other.type;
    runtime = other.runtime;
```

```
    // Copy genre array
    genreCount = other.genreCount;
    genre = new string[genreCount];
    for (int i = 0; i < genreCount; ++i) {
        genre[i] = other.genre[i];
    }

    // Copy actors array
    actorCount = other.actorCount;
    actors = new string[actorCount];
    for (int i = 0; i < actorCount; ++i) {
        actors[i] = other.actors[i];
    }

    country = other.country;
    rating = other.rating;
}

// Destructor to free dynamically allocated memory
~Entertainment() {
    delete[] genre;
    delete[] actors;
}

// Method to display information
void display() {
    cout << "Title: " << title << endl;
    cout << "Air Date: " << airDate << endl;
    cout << "Type: " << type << endl;
```

```
        cout << "Runtime: " << runtime << " minutes" << endl;
        cout << "Country: " << country << endl;

        cout << "Genres:";
        for (int i = 0; i < genreCount; ++i) {
            cout << " " << genre[i];
        }
        cout << endl;

        cout << "Actors:";
        for (int i = 0; i < actorCount; ++i) {
            cout << " " << actors[i];
        }
        cout << endl;

        cout << "Rating: " << rating << endl;
        cout << endl;
    }
};

int main() {
    // Create object obj1 and initialize
    string obj1_genre[] = { "Action", "Adventure", "Sci-Fi" };
    int obj1_genre_count = 3;
    string obj1_actors[] = { "Actor1", "Actor2", "Actor3" };
    int obj1_actor_count = 3;

    Entertainment obj1("Movie Title", "2023-01-01", obj1_genre, obj1_genre_count,
        "Movie", 120, "USA", obj1_actors, obj1_actor_count, 8.5);
```

```
    // Display obj1
    cout << "Object obj1:" << endl;
    obj1.display();

    // Create obj2 by copying only genre, country, and rating from obj1
    Entertainment obj2 = obj1;

    // Display obj2
    cout << "Object obj2 (copied from obj1):" << endl;
    obj2.display();

    return 0;
}
```

```
C:\Users\salma\Desktop\Ass5

Object obj1:
Title: Movie Title
Air Date: 2023-01-01
Type: Movie
Runtime: 120 minutes
Country: USA
Genres: Action Adventure Sci-Fi
Actors: Actor1 Actor2 Actor3
Rating: 8.5

Object obj2 (copied from obj1):
Title: Movie Title
Air Date: 2023-01-01
Type: Movie
Runtime: 120 minutes
Country: USA
Genres: Action Adventure Sci-Fi
Actors: Actor1 Actor2 Actor3
Rating: 8.5


--------------------------------
Process exited after 0.07889 seconds with return value 0
Press any key to continue . . .
```

Q2.

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

using namespace std;

struct Student {
   string name;
   int age;
   string registrationNumber; // Assuming registration number is a string
};

// Function to write student information to file
void writeStudentInfo() {
   ofstream fout("student.txt", ios::app); // Append mode
   if (!fout) {
      cerr << "Error opening file 'student.txt' for appending.\n";
      return;
   }

   Student student;
   cin.ignore();
   cout << "Enter student name: ";
   getline(cin, student.name);
   cout << "Enter student age: ";
   cin >> student.age;
```

```
    cin.ignore();
    cout << "Enter student registration number: ";
    getline(cin, student.registrationNumber);

    fout << student.name << "," << student.age << "," << student.registrationNumber << endl;

    cout << "Student information written to file successfully.\n";
    fout.close();
}

// Function to search for a student by name
void searchByName(const string& name) {
    ifstream fin("student.txt");
    if (!fin) {
        cerr << "Error opening file 'student.txt' for reading.\n";
        return;
    }

    string line;
    bool found = false;
    while (getline(fin, line)) {
        size_t pos1 = line.find(",");
        string student_name = line.substr(0, pos1);
        if (student_name == name) {
            size_t pos2 = line.find(",", pos1 + 1);
            int age = stoi(line.substr(pos1 + 1, pos2 - pos1 - 1));
            string regNum = line.substr(pos2 + 1);
            cout << "Student found:\n";
            cout << "Name: " << student_name << endl;
```

```
            cout << "Age: " << age << endl;
            cout << "Registration Number: " << regNum << endl;
            found = true;
            break;
        }
    }

    if (!found) {
        cout << "Student with name '" << name << "' not found.\n";
    }

    fin.close();
}

// Function to search for a student by age
void searchByAge(int age) {
    ifstream fin("student.txt");
    if (!fin) {
        cerr << "Error opening file 'student.txt' for reading.\n";
        return;
    }

    string line;
    bool found = false;
    while (getline(fin, line)) {
        size_t pos1 = line.find(",");
        string student_name = line.substr(0, pos1);
        size_t pos2 = line.find(",", pos1 + 1);
        int student_age = stoi(line.substr(pos1 + 1, pos2 - pos1 - 1));
```

```
        if (student_age == age) {
            string regNum = line.substr(pos2 + 1);
            cout << "Student found:\n";
            cout << "Name: " << student_name << endl;
            cout << "Age: " << student_age << endl;
            cout << "Registration Number: " << regNum << endl;
            found = true;
            break;
        }
    }

    if (!found) {
        cout << "Student with age '" << age << "' not found.\n";
    }

    fin.close();
}

// Function to search for a student by registration number
void searchByRegistrationNumber(const string& regNum) {
    ifstream fin("student.txt");
    if (!fin) {
        cerr << "Error opening file 'student.txt' for reading.\n";
        return;
    }

    string line;
    bool found = false;
    while (getline(fin, line)) {
```

```
        size_t pos1 = line.find(",");
        string student_name = line.substr(0, pos1);
        size_t pos2 = line.find(",", pos1 + 1);
        int age = stoi(line.substr(pos1 + 1, pos2 - pos1 - 1));
        string student_regNum = line.substr(pos2 + 1);
        if (student_regNum == regNum) {
            cout << "Student found:\n";
            cout << "Name: " << student_name << endl;
            cout << "Age: " << age << endl;
            cout << "Registration Number: " << student_regNum << endl;
            found = true;
            break;
        }
    }

    if (!found) {
        cout << "Student with registration number '" << regNum << "' not found.\n";
    }

    fin.close();
}

// Main here
int main() {
    int choice;
    string name;
    int age;
    string regNum;
```

```
do {
    cout << "\nOptions:\n";
    cout << "1. Write student information to file\n";
    cout << "2. Search student by name\n";
    cout << "3. Search student by age\n";
    cout << "4. Search student by registration number\n";
    cout << "5. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
    case 1:
        writeStudentInfo();
        break;
    case 2:
        cout << "Enter student name to search: ";
        cin.ignore();
        getline(cin, name);
        searchByName(name);
        break;
    case 3:
        cout << "Enter student age to search: ";
        cin >> age;
        searchByAge(age);
        break;
    case 4:
        cout << "Enter student registration number to search: ";
        cin.ignore();
        getline(cin, regNum);
```

```
        searchByRegistrationNumber(regNum);
        break;
    case 5:
        cout << "Exiting program.\n";
        break;
    default:
        cout << "Invalid choice. Please enter again.\n";
    }
  } while (choice != 5);


  return 0;
}
```

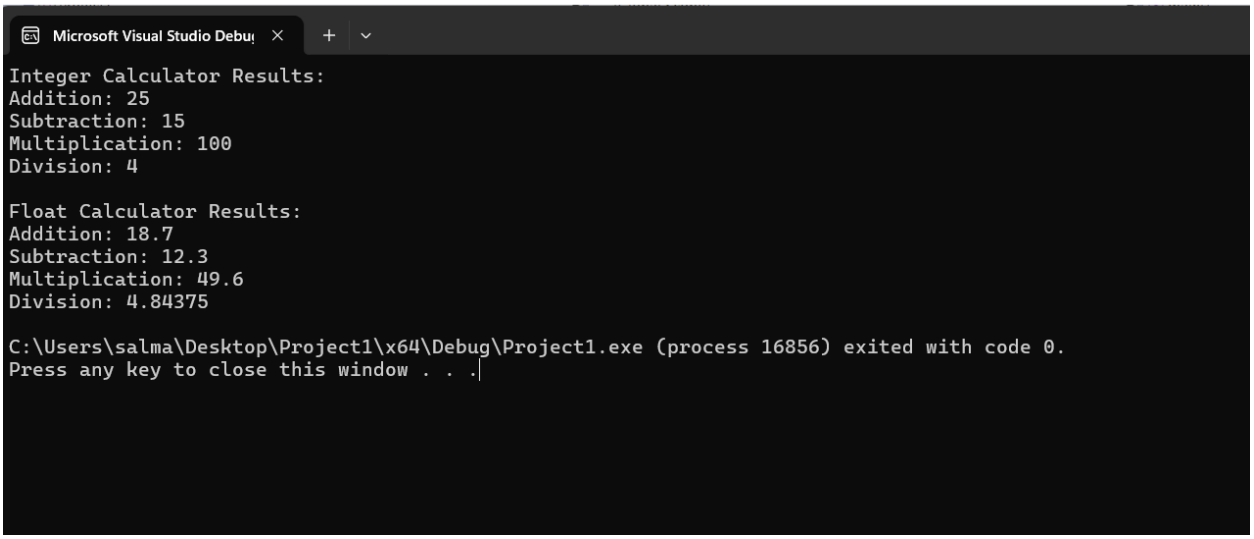
```
Microsoft Visual Studio Debu

Enter your choice: 1
Enter student name: Salman
Enter student age: 20
Enter student registration number: 050
Student information written to file successfully.

Options:
1. Write student information to file
2. Search student by name
3. Search student by age
4. Search student by registration number
5. Exit
Enter your choice: 4
Enter student registration number to search: 050
Student found:
Name: Salman
Age: 22
Registration Number: 050

Options:
1. Write student information to file
2. Search student by name
3. Search student by age
4. Search student by registration number
5. Exit
Enter your choice: 5
Exiting program.

C:\Users\salma\Desktop\Project1\x64\Debug\Project1.exe (process 24348) exited with code 0.
Press any key to close this window . . .
```

Q3.

```
#include <iostream>

using namespace std;

// Template class Calculator
template <typename T>
class Calculator {
private:
  T num1;
  T num2;

public:
  // Parameterized constructor
  Calculator(T n1, T n2) {
    num1 = n1;
    num2 = n2;
  }

  // Function to perform addition
  T addition() {
    return num1 + num2;
  }

  // Function to perform subtraction
  T subtraction() {
    return num1 - num2;
  }
```

```
    // Function to perform multiplication
    T multiplication() {
        return num1 * num2;
    }

    // Function to perform division
    double division() {
        if (num2 != 0) {
            return static_cast<double>(num1) / num2;
        }
        else {
            cout << "Error: Division by zero\n";
            return 0;
        }
    }
};

int main() {

    Calculator<int> intCalc(20, 5);

    cout << "Integer Calculator Results:" << endl;
    cout << "Addition: " << intCalc.addition() << endl;
    cout << "Subtraction: " << intCalc.subtraction() << endl;
    cout << "Multiplication: " << intCalc.multiplication() << endl;
    cout << "Division: " << intCalc.division() << endl;

    Calculator<float> floatCalc(15.5f, 3.2f);
```

```
    cout << "\nFloat Calculator Results:" << endl;
    cout << "Addition: " << floatCalc.addition() << endl;
    cout << "Subtraction: " << floatCalc.subtraction() << endl;
    cout << "Multiplication: " << floatCalc.multiplication() << endl;
    cout << "Division: " << floatCalc.division() << endl;

    return 0;
}
```



Q4.

```
#include <iostream>
#include <algorithm> // for std::sort
#include <string>

using namespace std;

// Template class Queue
template <typename T>
class Queue {
private:
  T list[10];

public:

  Queue() {
    for (int i = 0; i < 10; ++i) {
      list[i] = T();
    }
  }

  // Function to sort the elements of the queue
  void sort() {
    std::sort(list, list + 10);
  }

  // Function to find the maximum
  T max() {
```

```
    T maxValue = list[0];
    int count = 1;

    for (int i = 1; i < 10; ++i) {
        if (list[i] > maxValue) {
            maxValue = list[i];
            count = 1;
        } else if (list[i] == maxValue) {
            ++count;
        }
    }

    cout << "Maximum value " << maxValue << " found " << count << " times.\n";
    return maxValue;
}

//Function to Find minimum
T min() {
    T minValue = list[0];
    int count = 1;

    for (int i = 1; i < 10; ++i) {
        if (list[i] < minValue) {
            minValue = list[i];
            count = 1;
        } else if (list[i] == minValue) {
            ++count;
        }
    }
```

```
        cout << "Minimum value " << minValue << " found " << count << " times.\n";
        return minValue;
    }



    void return_queue() {
        cout << "Queue elements:";
        for (int i = 0; i < 10; ++i) {
            cout << " " << list[i];
        }
        cout << endl;
    }
};

int main() {
    //  Queue objects with different data types
    Queue<int> intQueue;
    Queue<double> doubleQueue;
    Queue<string> stringQueue;

    cout << "Integer Queue:" << endl;
    intQueue.return_queue();
    cout << "Sorting integer Queue..." << endl;
    intQueue.sort();
    intQueue.return_queue();
    cout << "Maximum value in integer Queue: " << intQueue.max() << endl;
    cout << "Minimum value in integer Queue: " << intQueue.min() << endl;
```

```
    cout << "\nDouble Queue:" << endl;
    doubleQueue.return_queue();
    cout << "Sorting double Queue..." << endl;
    doubleQueue.sort();
    doubleQueue.return_queue();
    cout << "Maximum value in double Queue: " << doubleQueue.max() << endl;
    cout << "Minimum value in double Queue: " << doubleQueue.min() << endl;



    cout << "\nString Queue:" << endl;
    stringQueue.return_queue();



    stringQueue.return_queue();
    cout << "Maximum value in string Queue: " << stringQueue.max() << endl;
    cout << "Minimum value in string Queue: " << stringQueue.min() << endl;

    return 0;
}
```

C:\Users\salma\Desktop\Ass5

```
Integer Queue:
Queue elements: 0 0 0 0 0 0 0 0 0 0
Sorting integer Queue...
Queue elements: 0 0 0 0 0 0 0 0 0 0
Maximum value 0 found 10 times.
Maximum value in integer Queue: 0
Minimum value 0 found 10 times.
Minimum value in integer Queue: 0

Double Queue:
Queue elements: 0 0 0 0 0 0 0 0 0 0
Sorting double Queue...
Queue elements: 0 0 0 0 0 0 0 0 0 0
Maximum value 0 found 10 times.
Maximum value in double Queue: 0
Minimum value 0 found 10 times.
Minimum value in double Queue: 0

String Queue:
Queue elements:
Queue elements:
Maximum value  found 10 times.
Maximum value in string Queue:
Minimum value  found 10 times.
Minimum value in string Queue:

--------------------------------
Process exited after 0.04224 seconds with return value 0
Press any key to continue . . .
```