Data Structure and Algorithms

Lab Journal - Lab 0

Name: Ishmam Asim

Enrollment #: 01-134232-079 Class/Section: 3D

Objective

This lab is intended to provide a recap of concepts in C++ and OOP that will be frequently used in Data Structure and Algorithms course.

Task 1: Exercises

Implement the following exercises.

Exercise 1 :

a) Declare a class named **House** for a real estate locator service. The following information should be included:

Owner: (a string of up to 20 characters)
Address: (a string of up to 20 characters)
Bedrooms: (an integer)
Price (floating point)

b) Declare **available** to be an array of 100 objects of class **House.**

c) Write a function to read values into the members of an object of **House**.

d) Write a driver program to test the data structures and the functions you have developed.

The driver program should read in house entries into the **available** array. After the code for entering the data, you should write code to output the data that you have entered to verify that it is correct.

Your program should look like this:

Enter Owner : *M. Khan*
Enter Address : *G-9, Islamabad*
Number of Bedrooms? : *4*

Price : *4500000*
Enter another house? N The output should look like:

Owner	Address	Bedrooms	Price
M. Khan	G-9, Islamabad	4	4500000

```
Enter owner: M.Khan
Enter Address: G-9,Islamabad
Number of bedroom?: 4
Price: 4500000
Enter another house?: N
Owner    Address          Bedrooms        Price
 M.Khan    G-9,Islamabad    4              4500000
```

```
#include<iostream>

using namespace std;

class House {

private:
	string Owner;
	string Address;
	int Bedrooms;
	float price;

public:
	House() {
		Owner = " ";
		Address = " ";
		Bedrooms = 0;
		price = 0.0;




	}

	void set_Owner(string n) {
		Owner = n;



	}

	void set_Address(string n) {
		Address = n;



	}

	void set_Bedroom(int n) {
		Bedrooms = n;
```

```
        }

        void set_Price(float n) {
                price = n;


        }

        string get_owner() {

                return Owner;


        }


        string get_Address() {

                return Address;


        }


        int get_Bedroom() {


                return Bedrooms;

        }

        int get_Price() {


                return price;

        }

};

void main() {

        House avilable[100];
        string a,choice;
        int b=0, count=0;
        float c = 0.0;
        bool con = true;

        while (con) {
                cout << "Enter owner: ";
                cin >> a;
                avilable[count].set_Owner(a);

                cout << "Enter Address: ";
```

```
            cin >> a;
            avilable[count].set_Address(a);

            cout << "Number of bedroom?: ";
            cin >> b;
            avilable[count].set_Bedroom(b);

            cout << "Price: ";
            cin >> c;
            avilable[count].set_Price(c);

            cout << "Enter another house?: ";
            cin >> choice;

            while (true) {
                  if (choice == "N" || choice == "n") {
                        con = false;
                        break;
                  }
                  else if (choice == "Y" || choice == "y") {
                        count++;  // Correct the count increment
                        break;
                  }
                  else {
                        cout << "Invalid choice. Enter 'Y' for Yes or 'N' for No:
";
                        cin >> choice;
                  }
            }
      }

      for (int i = 0; i <= count; i++) {

            cout << "Owner      Address      Bedrooms     Price\n ";
            cout << avilable[i].get_owner()<<"   ";
            cout << avilable[i].get_Address() << "   ";
            cout << avilable[i].get_Bedroom() << "             ";
            cout << avilable[i].get_Price() << "   ";









      }


}
```

Extra Credit:

The real estate company is very happy with the program that was developed in the earlier to track their listings. Now they want to add some features to the processing. Additional features:

Search for a house that meets a potential buyer's specifications for the following:

- The price is not more than a specified amount

- The size is not less than a specified number of bedrooms The house with lowest price
- The largest house (with maximum number of bedrooms)
- In a given city
- With best ratio price/size
- The user may enter a "?" to indicate no preference.

Print all the entries that meet the buyer's need.

```
Enter owner: SRK
Enter Address: MUMBI
Number of bedroom?: 4
Price: 30000
Enter another house?: Y
Enter owner: JFK
Enter Address: AMERICA
Number of bedroom?: 9
Price: 20000
Enter another house?: N
Enter maximum wanted price (? for no preference): 20000
Enter minimum wanted size (number of bedrooms) (? for no preference): 9
Enter wanted city (? for no preference): ?
Owner   Address         Bedrooms        Price
 JFK             AMERICA             9             20000
```

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

class House {

private:
	string Owner;
	string Address;
	int Bedrooms;
	float price;

public:
	House() {
		Owner = " ";
		Address = " ";
		Bedrooms = 0;
		price = 0.0;




	}

	void set_Owner(string n) {
		Owner = n;
```

```
        }

        void set_Address(string n) {
                Address = n;



        }

        void set_Bedroom(int n) {
                Bedrooms = n;



        }

        void set_Price(float n) {
                price = n;



        }

        string get_owner() {

                return Owner;


        }


        string get_Address() {

                return Address;


        }


        int get_Bedroom() {


                return Bedrooms;

        }

        int get_Price() {


                return price;

        }

};
void Find_house(House houses[], int size) {
        string found_price, found_size, found_city;
```

```
        float Max_price;
        int Min_size;
        string city;


        cout << "Enter maximum wanted price (? for no preference): ";
        cin >> found_price;

        cout << "Enter minimum wanted size (number of bedrooms) (? for no preference):
";
        cin >> found_size;

        cout << "Enter wanted city (? for no preference): ";
        cin >> found_city;


        if (found_price!="?" )
        {
                Max_price = stof(found_price);
        }
        else
        {
                Max_price = 999999999;

        }


        if (found_size != "?")
        {
                Min_size = stoi(found_size);
        }
        else
        {
                Min_size = -1;

        }

        if (found_city != "?")
        {
                city= found_city;
        }
        else
        {
                city="";

        }

        cout << "Owner      Address      Bedrooms     Price\n ";

        if (city != "")
        {
                for (int i = 0; i < 100; i++)
                {
                        if (houses[i].get_Price() <=
Max_price&&houses[i].get_Bedroom()>=Min_size&&houses[i].get_Address()==city)
                        {

                                cout << houses[i].get_owner() << "             ";
```

```
                        cout << houses[i].get_Address() << "              ";
                        cout << houses[i].get_Bedroom() << "              ";
                        cout << houses[i].get_Price() << "              ";



                }

            }




        }

        else
        {
            for (int i = 0; i < 100; i++)
            {
                if (houses[i].get_Price() <= Max_price && houses[i].get_Bedroom()
>= Min_size )
                {

                        cout << houses[i].get_owner() << "              ";
                        cout << houches[i].get_Address() << "              ";
                        cout << houses[i].get_Bedroom() << "              ";
                        cout << houses[i].get_Price() << "              ";



                }

            }


        }





}
void main() {

        House avilable[100];
        string a,choice;
        int b=0, count=0;
        float c = 0.0;
        bool con = true;

        while (con) {
            cout << "Enter owner: ";
            cin >> a;
            avilable[count].set_Owner(a);

            cout << "Enter Address: ";
            cin >> a;
```

```
            avilable[count].set_Address(a);

            cout << "Number of bedroom?: ";
            cin >> b;
            avilable[count].set_Bedroom(b);

            cout << "Price: ";
            cin >> c;
            avilable[count].set_Price(c);

            cout << "Enter another house?: ";
            cin >> choice;

            while (true) {
                  if (choice == "N" || choice == "n") {
                        con = false;
                        break;
                  }
                  else if (choice == "Y" || choice == "y") {
                        count++;  // Correct the count increment
                        break;
                  }
                  else {
                        cout << "Invalid choice. Enter 'Y' for Yes or 'N' for No:
";
                        cin >> choice;
                  }
            }
      }

      Find_house(avilable,100);




}
```

Exercise 2:

Assume that a file contains the midterm1, midterm2 and final exam scores and names of students of a class. Write a C++ program to read the input file and produce an output file containing the original and average scores for each student. Suppose that the weights of the exams are as follows:

midterm1 – 25%
midterm2 – 25% final
– 50%.

The average score of a student is calculated using the formula:

$0.25*MT1 + 0.25*MT2 + 0.5*FIN$

Students.txt

File Edit View

```
Ali 85 90 80
Hamad 78 82 88
Ahmad 90 85 95
```

Ln 4, Col 1 | 43 characters | 100% | Windows (CRLF) | UTF-8

Students.txt StudentsWithAverages.txt

File Edit View

```
Name    Midterm1        Midterm2        Final   Average
John    85      90      80      83.755
Jane    78      82      88      84.005
Alice   90      85      95      91.255
```

Ln 1, Col 1 | 101 characters | 100% | Windows (CRLF) | UTF-8

```
#include <iostream>
#include <fstream>
```

```
#include <string>

using namespace std;

int main()
{
    ifstream inputFile("Students.txt");
    ofstream outputFile("StudentsWithAverages.txt");
    string name;
    double midterm1, midterm2, finalExam;

    if (inputFile.is_open())
    {

        if (outputFile.is_open())
        {
            outputFile << "Name\tMidterm1\tMidterm2\tFinal\tAverage" << endl;

            while (inputFile >> name >> midterm1 >> midterm2 >> finalExam)
            {
                double average = 0.25 * midterm1 + 0.25 * midterm2 + 0.5 *
finalExam;


                outputFile << name << "\t"
                    << midterm1 << "\t"
                    << midterm2 << "\t"
                    << finalExam << "\t"
                    << (average * 100 + 0.5) / 100.0 << endl;


            }

        }
        else
        {
            cout << "Unable to open output file." << endl;
            return 1;
        }


    }
    else
    {
        cout << "Unable to open input file." << endl;
        return 1;
    }


    inputFile.close();
    outputFile.close();



    return 0;
}
```

Exercise 3:

You will write a student grades "database" program. It will read data of students from a file and will let the user perform various operations on the data. You will have to store the student data in an array of objects.

Input:
The input file will look like:
4
3
Hassan Khan 99 87 90
Sara Nazir 90 98 99
Ali Zaidi 55 43 0
Raza Ahmad 100 100 100

That is:
number of students number of
grades (per student) Student name
grade grade ... grade
Student name grade grade ... grade

Data structure :

You will store all the information in an array of "student" objects. You may use the following class definition:

```
class student {
private:
    char name[30];
    int lab[10]; float
    average;
public:
     //Any functions you want to create
};
```

```
Name: Hassan Khan
Lab Scores: 99 87 90
Average: 92
Name: Sara Nazir
Lab Scores: 90 98 99
Average: 95.6667
Name: Ali Zaidi
Lab Scores: 55 43 0
Average: 32.6667
Name: Raza Ahmad
Lab Scores: 100 100 100
Average: 100
```

```
#include <iostream>
#include <fstream>

using namespace std;

class student {
private:
    char name[30];
    int lab[10];
    float average;
public:
    void set_name(const char* n) {
        int i = 0;

        // Copy characters from n to name until null terminator or size limit
        while (n[i] != '\0' && i < 29) {
            name[i] = n[i];
            i++;
        }

        // Null-terminate the destination string
        name[i] = '\0';
    }

    void set_lab(int *n,int size) {
        for (int i = 0; i < size; i++) {
            lab[i] = n[i];
        }
    }

    void set_avg(int size) {
        int total = 0;

        for (int i = 0; i < size; i++) {
            total = total + lab[i];
        }
        average = static_cast<float>(total) / size;
    }


    void get_name() {
        cout << "Name: " << name << endl;
```

```
    }

    void get_lab(int size) {
        cout << "Lab Scores: ";
        for (int i = 0; i < size; i++) {
            cout << lab[i] << " ";
        }
        cout << endl;
    }

    void get_average() {
        cout << "Average: " << average << endl;
    }
};

void combineArrays(const char* array1, const char* array2, char* result, size_t
result_size) {
    size_t i = 0;
    size_t j = 0;

    // Copy array1 to result
    while (array1[i] != '\0' && i < result_size - 1) {
        result[i] = array1[i];
        i++;
    }

    // Add a space or delimiter between arrays (optional)
    if (i < result_size - 1) {
        result[i] = ' ';
        i++;
    }

    // Copy array2 to result
    while (array2[j] != '\0' && i < result_size - 1) {
        result[i] = array2[j];
        i++;
        j++;
    }

    // Null-terminate the result array
    result[i] = '\0';
}

int main() {
    ifstream file("data.txt");
    int n = 0, m = 0,count=0;
    student* s=new student[n];

    int* score = new int[m];
    char fname[30], lname[30], fullname[30];

    if (file.is_open()) {
        file >> n;
        file >> m;

        while (file >> fname >> lname)
        {
            combineArrays(fname, lname, fullname, 30);
```

```
            s[count].set_name(fullname);
            for (int i = 0; i < m; i++)
            {
                file >> score[i];

            }
            s[count].set_lab(score,m);
            s[count].set_avg(m);

            s[count].get_name();
            s[count].get_lab(m);
            s[count].get_average();

        }
    }
    else {
        cout << "Error opening file." << endl;
    }

    return 0;
}
```

Implement the given exercises and get them checked by your instructor.

S No.	Exercise	Checked By:
1.	Exercise 1	
2.	Exercise 2	
3.	Exercise 3	

++++++++++++++++++++++++++