CSCI 2170 Final Exam review

Topics Covered:

1. One dimensional array
    a. Read data into array and record the number of items in the array
    b. Use array subscript to access array elements
    c. Traversal in an array
    d. Linear search in an array
    e. Binary search in an array
    f. Sorting in array
    g. Insert and delete items into/from array
2. Two dimensional array
    a. Use nested for loop with two dimensional array
    b. be able to perform operations (i.e., computing the average) along one column of the 2 dimensional array, or along one row of the 2 dimensional array
    c. be able to perform operations (i.e., search for the smallest value) across all the elements (i.e., across all the rows and columns) of the 2 dimensional array
    d. look over the class notes and practice questions at the end of the class notes
3. Struct type
    a. Define a struct type.
    b. Declare variable of struct type
    c. Use dot notation to access members of a struct type variable
    d. Use array of structs to maintain a list of records (OLA4)
        i. Read records of information from a data file and store in an array of structs, record the number of records read
        ii. Print records
            1. Print records that satisfy certain condition (i.e., all books written by the same author)
        iii. Search for records that satisfy certain condition
        iv. Sort records by along certain feature, i.e., sort books in ascending order by author name

4. Pointer, dynamic memory allocation

- How to declare pointer variable
- Use pointer variable to point to memory locations
- Use new operator to acquire memory during run time
- Use delete operator to release memory
- **Go over the exercises discussed in class**

5. Linked list related questions

Define a node structure for linked list
Create a new node using dynamic memory allocation
Free the memory of a node
List traversal:

- Print all the values stored in a linked list
- Compute the sum of all the values in a linked list
- Find whether a value is in the list
- Update a value in the linked list to a new value

Insertion operation

- Insertion always happen at the front of the list
- Insertion always happen at the end of the list
- Insertion by location

- Insertion into sorted list and maintain the list to be sorted after insertion

Deletion operation

- Deletion by location
- Deletion from a sorted list and maintain the list to be sorted after deletion

Build a list by read data from a data file

Destroy a list

Make a deep copy of an existing list

6. Abstract Data Type (ADT) related questions:

- Be able to define a ADT by defining its header file and specification file
- Be able to write client program using the ADT defined above
- ***Be able to write a client program using ADT list (array implementation or linked list implementation)***
- ***Understand how the array based list ADT is defined.***
    - ***Be able to add new methods by defining the method in the header file and define the method in the implementation file***
- ***Understand how the linked list based list ADT is defined***
    - ***Be able to add new methods by defining the method in the header file and define the method in the implementation file***

(Topics discussed after Test 3)

Example Questions

1. Construct a linked list based on user inputs (from keyboard, or from file)
2. Show function to traverse the linked list (i.e., print data from all the nodes)
3. Show function to add data to a linked list
4. Show function to delete a value from a linked list

5. A complex number consists of two components: the real component and the imaginary component. An example of a complex number is 2+3i, where 2 is the real component and 3 is the imaginary component of the data. Define a class MyComplexClass. It has two data values of float type: **real** and **imaginary.**

This class has the following member functions:

- A **default constructor** that assigns 0.0 to both its real and imaginary data members;
- The **value constructor** that assigns client supplied (real and imaginary) values to the real and imaginary data members;
- The copy constructor
- Define the **accessor** and the **mutator** functions, for example
    - A **member function "SetValues"** that assigns client supplied values to the real and imaginary data members; (This is not a constructor);
    - A member function "**GetReal**" that returns the real component of the number;
- A member function "**Display**" that outputs the complex number in the form "a + bi" on screen, where a and b are the real and imaginary components.
- A **member function "EqualTo"** that compares two complex numbers. It returns true if they are the same, and returns false if they are different. Two complex numbers are considered the same if the real components of the two values are the same and the imaginary components of the two values are also the same.
- Call method to display the complex objects
- Overloaded < operator, overloaded == operator
- Write user defined functions that pass complex objects by value or by reference

<u>You are required to:</u>

(a) Write the complete header file for MyComplexClass;

(b) Write the complete implementation file for MyComplexClass.
(c) Write the client program to:

- Create two objects of MyComplexClass. One objects should be created using the default constructor, and the other with the value constructor;
- Use "Set" methods to change the first object to the complex number 5.5+3i
- Use "Display" method to display the first object
- Apply **EqualTo** function to compare the two complex numbers and output appropriate messages concerning whether the two numbers are the same or not.
- Declare the third complex number as a copy of the second complex number, ie., using copy constructor
- Write a user defined functions to
    - Add two complex numbers. For two complex numbers: a+bi and c + di, the addition of the two numbers is: (a+c) + (b+d)i
- Declare an array of 20 complex numbers
- Use overloaded operator to compare two MyComplexClass objects
- Write a user defined functions to
    - Assign the values of each of these 20 complex numbers
    - Display each of these 20 complex numbers

6. For this problem, you are required to use the **unsorted list class array implementation** as discussed in class. Write a C++ **client program** to read in a number of integer values from a data file and store the numbers in a list. The values are stored one value per line in the data file. After all the values are inserted into the list, call user-defined function "**AboveAverage**" to compute and return the number of values in the list that are above the average of all the values in the list.

```
// Assume all the needed header files have already been included properly
 // Declare the user-defined function here


int main()  {
    int count, data;   ifstream myIn("data");
    // declare a list class object here


    // read values from the data file and add to the list, repeat til the end of the data file is reached


    // call "AboveAverage" function to compute and return the number of values in the list that are
above the average value.
```

```
    cout << "The number of values above average is: " << count << endl;
    return 0;
}
```

// define the function "AboveAverage" here