CSCI 2170 Test 2 topics

- **Go over code went over in class, in closed and open labs**

- Topics covered:

  - Two dimensional array

  - Struct type
    - Define a struct type.
    - Declare variable of struct type
    - Use dot notation to access members of a struct type variable
    - Use array of structs to maintain a list of records
      - Read records of information from a data file and store in an array of structs, record the number of records read
      - Print records
        - Print records that satisfy certain condition (i.e., all books written by the same author)
      - Search for records that satisfy certain condition
      - Sort records by along certain feature, i.e., sort books in ascending order by author name

- Recursion
  - Be able to trace the execution of recursive function
  - Be able to write recursion function given recurrence relation

- **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
  - Use pointer to dynamically allocate memory for 1D and 2D array. Release memory for array acquired dynamically.

- **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
    - Delete from an unsorted list

- Deletion from a sorted list

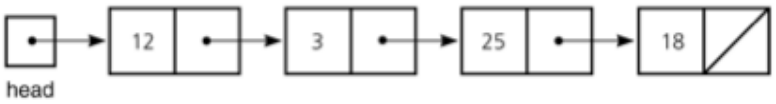
- **Abstract Data Type (ADT) related questions:**
  - Be able to define a ADT by defining its header file and specification file
    - Define constructors, accessor methods, mutator methods, operators that support comparison
    - Define overloaded operators
  - Be able to write client program using 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
  - **Be able to write client program using the ADT list**

Some Example test questions:

1. Go over the example code discuss in class on the topic of 2 dimensional array
2. The following code was written to print all the data in the linked list that has a value greater than 5. But the program ran into "segmentation fault" run time error. What is the problem with the code? Show how you would fix the code.

```
NodePtr cur;
cur=head;
while (cur→data>5 && cur!=NULL) {
        cout << cur→data << endl;
        cur = cur→ next;
}
```

3. Consider the **unsorted linked list** of integers as shown in the figure below:



   a. write the C++ statements that insert a new node that contains a value 9 in front of the node with value 12
   b. write the C++ statements that insert a new node that contains a value 10 after the node with value 18
   c. write the C++ statements that insert a new node that contains a value 5 into the list so that it is in between the nodes having values 12 and 3, i.e., the new node will be the 2nd node after insertion.
   d. Delete the node with value 25
   e. Delete all the nodes in the entire linked list

4. Linked list operations:
   f. Read values from a data file and build a linked list of these values
   g. Write the C++ function that will print all the values in a linked list
   h. Write the C++ function that will insert a value at the front of the linked list
   i. Write the C++ function that will insert a value at the end of the linked list
   j. Write the C++ function that will insert a value at the kth position of the linked list, k is a parameter supplied

k. Write the C++ function that will delete a value at the front of the list
l. Write the C++ function that will delete a given value from the list
m. Write the C++ function that will delete a value at the kth position of the list

1. 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