Homework 2 (points 100)

Due Date: on elearning

Submit: eLearning

Late Policy: 10% points per day late

Instructions: Include Name and Net ID. This is an individual assignment. Answers should be your own work.

1. Write pseudocode to implement a queue using 2 stacks. (enqueue, dequeue, Qfull, Qempty, display) [10 points]

```
Create a new empty stack1
Create a new empty stack2
Enqueue:
    If Qfull returns true
        Return saying that the queue is full
    Else
        Push the received data into the stack1
Dequeue:
    If Qempty returns true
        Return saying that the queue is empty
    If stack2 is empty
        While stack1 isn’t empty
            Pop from stack1 and push that data into stack2
    Return the a popped value from stack2

Qfull:
    If the size of stack1 plus the the size of stack 2 equals the maximum size
```

```
        Return true
    Else
        Return false
Qempty:
    If both stack1 and stack2 are empty
        Return true
    Else
        Return false
Display:
    If Qempty is true
        Return saying the queue is empty
    If stack 2 is empty
        While stack1 is not empty
            Pop stack1 value and push that data into stack2
    Using a loop, for each value in stack 2
            Print out that value
```

2. Write pseudocode to implement a stack using doubly linked list. (push, pop, stackEmpty, display, top) [10 points]

```
Create a node class which contains variables for the data, nextNode, and prevNode
Create a top and bottom variables to store the top and bottom nodes.
Push:
    Create a new node and store the received data into it
    If the stackEmpty is true
        Set top equal to this new node
        Set the bottom equal to this new node
    Else
        Set the prevNode of new node equal to top node
        Set the nextNode of  top node equal to new node
        Set the top equal to the new node
Pop:
    If stackEmpty is true
        Return saying the stack is empty
    Else
        Set top equal to the prevNode of top node
        If top node is null
            Set the bottom node equal to null
        Else
            Set top node’s nextNode equal to null
stackEmpty:
    If the top node is equal to null
        Return true
    Else
        Return false

Display:
    If stackEmpty is true
        Return saying the stack is empty
    Else
        Create a traversal node and set it equal to the top node
        While the traversal node is not null
            Print out the data of the node
            Set the traversal node equal to the nextNode of traversal node
Top:
    If stackEmpty returns true
        Return saying that the stack is empty
    Else
        Return the data contained within top node
```

3. Write pseudocode to implement a queue using doubly linked list. (enqueue, dequeue, Qempty, display) [10 points]

```
Create a node class with a prevNode variable, variable which contains data, and
nextNode variable

Create a variable to store the front node
Create a variable to store the rear node

Enqueue:
    Create a new node and store the received data into it
    If Qempty is true
        Set the front node equal to the new node
    Else
        Set the nextNode of the rear node equal to the new node
        Set the prevNode of the new node equal to the rear node
    Set the read node equal to the new node

Dequeue:
    If Qempty is true
        Return saying that the queue is empty
    If the front node is equal to the read node
        Set the front node to null
        Set the read node to null
    Else
        Set the front node equal to the nextNode of front node
        Set the prevNode of new front node to null

Qempty:
    If front node is equal to null
        Return true
    Else
        Return false

Display:
    If Qempty is true
        Return saying the queue is empty
    Create a traversal node and set it equal to the front node
    While the traversal node is not equal to null
        Print the data contained within the traversal node
        Set the traversal node equal to the next node of traversal node
```

4. Write pseudocode to implement a circular queue using doubly linked list. (enqueue, dequeue, Qempty, display) [10 points]

```
Create a node class with a prevNode variable, variable which contains data, and
nextNode variable

Create a variable to store the front node
Create a variable to store the rear node

Enqueue:
    Create a new node and store the data received into it
    If Qempty is true
        Set the front node equal to the new node
        Set the rear node equal to the new node
    Else
        Set the nextNode of rear node equal to the new node
        Set the new node’s prevNode equal to the rear node
        Set the front node’s prevNode equal to the new node
        Set the nextNode of the new node equal to the front node
        Set the rear node equal to the new node

Dequeue:
    If Qempty is true
        Return saying that the circular queue is empty
    If the front node is equal to the rear node
        Set the front node to null
        Set the rear node to null
    Else
        Set the front node equal the nextNode of front node
        Set the prevNode of front node equal to rear node
        Set the next node of rear node equal to front node


Qempty:

    If the front node is equal to null

        Return true

    Else

        Return false


    Display:

    If Qempty is true

        Return saying the circular queue is empty
    Create a traversal node and set it equal to the front node
    While the traversal node is not equal to the rear node
        Print out the data contained within the traversal node
        Set the traversal node equal to the nextNode of traversal node
    Print out the rear node's data
```

5. In some circumstances, the normal FIFO operation of a queue may need to be overridden This may occur due to priorities that are associated the elements of the queue that affect the order of processing. In cases such as these, a ***priority queue*** is used, where the elements are removed based on priority. How will you implement a priority queue? (enqueue, dequeue, Qempty, display) [10 points]

```
Create a PriorityQueue class using generic that extends to the Comparable class
Set up a head node

Create a node class which can store data received and store where the nextNode is

Enqueue:
    Create a new node and store the received data within it
    If Qempty is true OR the new node's data is less than head's data
        Set the nextNode of new node equal to the head node
        Set the head node equal to new node
    Else
        Create a traversal node and store the head node
        While the traversal node's nextNode is not equal to null AND the new
        node's data is greater than the traversal node's next node's data.
            Set traversal node equal to the traversal node's next node
        Set the new node's nextNode equal to the traversal node's nextNode
        Set the traversal node's nextNode equal to the new node.

Dequeue:
    If Qempty returns true
        Return saying the the priority queue is empty
    Set the head node equal to the nextNode of head node
    Return the data contained within the original head node


Qempty:
    If head is equal to null
        Return true
    Else
        Return false
```

Display:

```
    Create a traversal node and store the head node in it
    While the traversal node is not equal to null
        Print out the data of traversal node
        Set the next node of traversal node equal to the nextNode of traversal node
```

6. Write pseudocode to swap two adjacent elements by adjusting only the links (and not the data) using following data structures [10 Points]
   a. Singly linked list

```
Swap(head node, node1, node2)
    If node1 and node2 are equal OR nextNode of node1 is not equal to node 2
        Return
    If node 1 is equal to the head node
        Set the head node equal to node2
    Else
        Create a prevNode node and set it equal to the head node
        While the prevNode's nextNode is not equal to node1
            Set the prevNode equal to the next node of prevNode
        Set the prevNode's next node equal to node2
    Set the nextNode of node1 equal to the nextNode of node2
    Set the nextNode of node2 equal to node1
    Return headNode
```

   b. Doubly linked list

```
Swap(head, node1, node2)
```

```
If node1 is equal to node2 OR nextNode of node1 is not equal to node2 OR
prevNode of node 1 is null OR nextNode of node 2 is null
    Return
If node1 is equal to the head node
    Set the head node equal to node2
Else
    Set node1's prevNode's nextNode equal to node2
Using node2's next node, set the prevNode equal to node1
Set node1's next node equal to the next node of node2
Set node2's prevNode equal to the prevNode of node1
Set the nextNode of node2 equal to node1
Set prevNode of node1 equal to node2
Return headNode
```

7. Write pseudocode to implement the 'contains' routine for LinkedList program which checks if a linked list contains a given integer. [5 Points]

```
Contains(data)
Create a traversal node and store the head node into it
While the traversal node is not equal to null
    If the traversal node's data contains same data
    Return true
Return false
```

8. The Josephus problem is the following game: N people, numbered 1 to N, are sitting in a circle. Starting at person 1, a hot potato is passed. After M passes, the person holding the potato is eliminated, the circle closes ranks, and the game continues with the person who was sitting after the eliminated person picking up the hot potato. The last remaining person wins. Thus if M =0 and N=5, players are eliminated in order and player 5 wins. If M=1 and N=5, the order of elimination is 2,4,1,5.
   a. Write a pseudocode to solve the Josephus problem for general (integer) values of M and N. Try to make your program as efficient as possible. Make sure you dispose of cells.

```
Josephus(int N, M):

Create an array list of integers to store the people represented by integers

For each value until N
    Using arrayList add method, add number starting from 1
    Increment number for each loop

Create a current integer which will represent the person that is being processed
on in the arrayList
While the arrayList size is bigger than 1
    Set current to (current + M – 1) modulus of the current size of array list
```

```
        Remove the element using 'current' as the index
        Set current to the current modulus of arrayList size
    Return the first index of arrayList also known as the last person remaining
```

[7 Points]

b. What is the running time of your program?

The loop will run N-1 times and each iteration of the loop involves moving N-1 people and removing 1 person. Thus resulting in the running time of the program being O(N*M).

[3 Points]

9. What is the running time of the following code? [5 Points]

```
public static List<Integer> makeList( int N)

{

    ArrayList<Integer> lst = new ArrayList<>();

    for( inti  =0; i < N; i++)

    {

        lst.add(i);

        lst.trimToSize();

    }

}
```

```
The running time of the code is O(n^2) due to the for loop running N
times and the nested loop in trimToSize running possible n times
```

10. The following routine removes the first half of the list passed as a parameter: [10 Points]

```
public static void removeFirstHalf(List<?> lst)

{

    int theSize = lst.size() /2

    for( inti =0; i < theSize; i++ )

        lst.remove(0);

}
```

a. Why is theSize saved prior to entering the for loop?
   theSize is saved prior before entering the loop is because the objective of the loop is to remove the first half of the list. The list will only iterate to half way point of the size of the array.
b. What is the running time of removeFirstHalf if lst is an ArrayList? The loop will iterate n/2 times but we don't care about coefficients so it will run n times. The remove method for array list contains a for loop which searches for the element that needs to be removed so that will also run n times. The running time is therefore O(n^2).
c. What is the running time of removeFirstHalf if lst is a LinkedLIst?
   The running time of the method will be O(n) because removal in linked list is constant time. Wherever the iterator is at, remove method can remove at the iteration without having to loop through the linked list.
d. Does using an iterator make removeFirstHalf faster for either type of List ?
   No it does not. The array list is still O(n^2) and the linked list is still O(n).

11. Write a function in pseudocode named removeDuplicates(), which takes a singly linked list sorted in increasing order and deletes any duplicate nodes from the list. The list should only be traversed once and the routine should not call any other routine.

For example if the linked list is 11->11->11->21->43->43->60 then removeDuplicates() should convert the list to 11->21->43->60.

Input: the **head** node of the linked list .
Output:
Your function should return a pointer to the head of linked list with no duplicate element.

```
RemoveDuplicates(headNode):
    If head node is null or nextNode of head node is null
        Return head node
    Create a traversal node and store the head node in it
    While the traversal node's nextNode is not equal to null
    If the traversal node's data is equal with the nextNode of traversal node
        Set the nextNode of traversal node equal to the next nextNode of traversal node
  Else
        Set the traversal node to the nextNode of traversal node
  Return head node
```