Assignment 2
(Due Friday of week 5 - Nov 30)

1. Write separate programs for the following exercises in Java, Python, C#, or C++. Each file should have your name at the top in comment, with short description of what is implemented in that file. Make sure your files have appropriate names as indicated in each exercise. Programs should write output to the Console and have hard coded input in main.

   Note 1: If a program is not in approved programming language (or in different language than previous assignment) or has any syntax error, no points will be awarded for that exercise

   Note 2: Submitting wrong files or in the wrong format or corrupted files will not be accepted nor will any re-submission be allowed for any such mistake. It is your responsibility to submit the correct files.

   a) **DLinkedListADT:** Implement doubly linked list ADT using the pseudo code covered in class. You should have all the operations in the pseudo code. You can use int for data-type. Write test program with main method creating an instance of the DLinkedListADT and calling all the operations to demonstrate they work correctly. You must implement a class called DLinkedListADT and separate TestDLinkedList class each in a separate file.

   b) **DeleteFirstInstance:** Add a new operation called deleteFirstInstance to the above DLinkedListADT which takes a "data-type num" parameter (you can use "int num" if your ADT is coded for int) and deletes the first node in the list where the data matches num.

      Write test program with main method to create and populate the DLinkedListADT instance with some elements, display the contents of the list, call the deleteFirstInstance method, and then print the modified list contents. You must have a test class called TestDeleteFirstInstance with main method.

      Evaluate and report the time and space complexity of your algorithm in a document or as comments in code.

      Run the program with the list populated with "12 34 65 76 23 22 36 90" in that order and following values for deleteFirstInstance:

      q.deleteFirstInstance(10) // no such element so list still "12 34 65 76 23 22 36 90"
      q.deleteFirstInstance(12) // first element removed "34 65 76 23 22 36 90"
      q.deleteFirstInstance(90) // last element removed "12 34 65 76 23 22 36"
      q.deleteFirstInstance(76) // 4th element removed "12 34 65 23 22 36 90"

      Example Output:

      ```
      Original list elements: 12 34 65 76 23 22 36 90
      num=76
      ```

```
Modified list elements: 12 34 65 23 22 36 90
```

c) **QueueADT**: Implement raw array **Queue ADT** to include operations:
- QueueADT(int capacity)
- boolean isEmpty()
- boolean isFull()
- void enQueue(int data)
- int deQueue()
- int size()

You must use the pseudo code covered in class. You should not have any other operations. Write test program with main method to demonstrate the correct working of each operation. You must implement a class called QueueADT and separate TestQueue class each in a separate file.

d) **StackADT**: Implement **StackADT** using the DLinkedListADT from previous exercise to include operations:
- StackADT(int capacity)
- boolean isEmpty()
- boolean isFull()
- void push(int data)
- int pop()
- int size()

Yu must use the pseudo code covered in class. You should not have any other operations. Write main method to demonstrate the correct working of each operation in a separate test program. You must implement a class called QueueADT and TestQueue class in a separate file.

e) **QueueReverseMiddle:** Write an algorithm, implemented as a method queuReverseMiddle that takes as input an instance of QueueADT (use the above implementation of QueueADT) and two integers: low and high.

Then it uses StackADT and/or QueueADT (use the above implementation of StackADT and QueueADT) to reverse the order of the low to high inclusive elements in the QueueADT instance. Your algorithm should throw an exception when passed in queue instance is empty or either low or high out of range.

You can only use the StackADT operations and QueueADT operations listed above to implement the solution and you cannot use recursion. Main method should create and populate a QueueADT instance, call the queueReverseMiddle algorithm, and display the changed QueueADT instance contents. You must implement a single class called QueueReverseMiddle with the method queueReverseMiddle and main method. Evaluate and report the time and space complexity of your algorithm in a document or as comments in code. You can earn 5 extra points for making algorithm time efficient.

Run the program with the queue populated with “12 34 65 76 23 22 36 90” in that order and following values for queueReverseMiddle:

queueReverseMiddle(q, 0, 8) // should fail with exception and message that invalid input
queueReverseMiddle(q, 1, 8) // whole queue is reversed “90 36 22 23 76 65 34 12”
queueReverseMiddle(q, 2, 8) // 2 through 8 reversed “12 90 36 22 23 76 65 34”
queueReverseMiddle(q, 3, 6) // 3 through 6 reversed “12 34 22 23 76 65 36 90”

Output example (red shows what was changed - just here not in the program):

```
low=3 high=6
Original Queue contents:  12 34 65 76 23 22 36 90
New Queue contents:       12 34 22 23 76 65 36 90
```

2. Record a video 10-15min long explaining the implementation and solution of each of the above programs to include showing the running program and output.

3. **Submission instructions:** Submit one zip file with all the programs and another file with the video

Grading Rubric

NOTE: You will not earn any points for the programs unless you submit the video which explains them.

Points	Criteria
10	Programs use object oriented program approach (e.g. class for Queue, Stack, etc.). Programs have the appropriate naming convention, author’s name, and brief description of the implementation in each file
15	**DLinkedListADT:** Correctly implements DLinkedListADT and its operations using pseudo code covered in class Correctly implements the test program TestDLinkedList to demonstrate correctness of the code Video correctly explains ADT and test implementation, and shows and explains running program and output
25	**DeleteFirstInstance:** Correctly adds method to DLinkedListADT called deleteFirstInstance(data-type num) Correctly implements algorithm to delete the first node that matches num Correctly loads data and calls the algorithm in test program’s main method Video correctly explains algorithm implementation, shows and explains running program and output Video correctly explains the time and space complexity of the deleteFirstInstance algorithm

30	Correctly implements raw array **QueueADT** in single file using pseudo code covered in class Correctly implements linked list implementation of **StackADT** in single file using **DLinkedListADT** and pseudo code covered in class Video correctly explains implementation and operations of each ADT and shows the test programs running to demonstrate operations
20	**QueueReverseMiddle:** Correctly only uses QueueADT and StackADT and its operations Correctly implements algorithm as method queueReversemiddle to reverse low to high element inclusively in passed in Queue instance Correctly writes output as above to include updated queue contents in main method only using valid Queue operations Video correctly explains the algorithm, implementation, shows and explains running program and output Video correctly explains the time and space complexity of the queueReverseMiddle algorithm