There are 1 questions for a total of 100 points.

---

(100) 1. The second programming assignment will involve using one data structure to implement another.

In lectures, we discussed implementation of Dynamic Arrays using Arrays by doubling the size of the array on overflows (the only operation supported for this Dynamic Array was insertion.) In this assignment, you are asked to implement Dynamic Arrays that support operations other than insert. You are given an outline of this class which you have to complete. This class is named `dynamicArray` and the description of this class is given using appropriate comments in the file `dynamicArray.java`.

Task 1: Complete the code for `dynamicArray.java`. Make sure that you only complete the methods that are given in the file. Please do not add new methods in this file.

Once you have implemented dynamicArray, you are now asked to implement a stack and a queue using this dynamic array. The outline for this is given in files `myStackUsingDynamicArray.java` and `myQueueUsingDynamicArray.java`. You may introduce private fields and methods (other than those given) in these files as per your convenience. Your implementation (for both stack and queue) should be space-efficient. This means that if the number of elements currently in the stack/queue is n, then the current size of the array used in the class `dynamicArray` should not be more than $2n$. This can be ensured by appropriately using the `doubleSize()` and `halveSize()` methods. Also, your implementation should be such that a sequence of n push operations for Stack (and n enqueue operations for Queue) should cost $O(n)$ time.

Task 2: Complete `myStackUsingDynamicArray.java` and `myQueueUsingDynamicArray.java`.

We would also implement a stack and a queue using a singly linked list. One issue with the previous implementation of Stack and Queue using dynamic array was that the type of data that can be stored in the stack or queue was determined by the type of array that was defined in the class `dynamicArray`. This happened to be *integer*. So, we cannot use the stack for storing Strings for instance. Generics in Java provide a neat solution to this issue by allowing a programmer to define the type of the data as a parameter when defining a class. Please go through Section 2.5 in the textbook to know more about java generics. The Singly Linked List implementation given in Section 3.2.1 (code fragment 3.13 and 3.14) of the book uses generics to define a linked list that can store any type of data. We will use this implementation of singly linked list for this assignment. This means that you will have to type up the code given in the book as it is. You are not allowed to make any changes in this class. Using this implementation of linked list you will have to implement a Stack and Queue. The outline of these are given in files `myStackUsingLinkedList.java` and `myQueueUsingLinkedList.java`.

Task 3: Complete `myStackUsingLinkedList.java` and `myQueueUsingLinkedList.java`..

Evaluation: Evaluation of homework consists of the following two components:

1. Submission component (75 points): Your code will be checked for correctness and running time. This will partially be done using an automated scripts. So, please make sure that you strictly follow these instructions:
   - Write all your code in a directory named ⟨Your entry number⟩.
   - For submission, create a zip file named ⟨Your entry number⟩.zip of your directory and submit. Details of where to submit will be sent in some time.
2. Viva component (25 points): We will hold all the labs during an evaluation week and you will be expected to attend the lab. During the lab, the following will be done:
   - You will be asked to make a small addition in your program to check your understanding.