COS 226 Programming Assignment

Randomized Queues and Deques

Write a generic ADT for a randomized queue and for a deque. The goal of this assignment is to implement elementary data structures using arrays and linked lists, and to get you reacquainted with programming in Java.

Install Java. Install Java for your operating system. Note that Java 1.5 is required for generics. [Windows · Mac OS X · Linux]

Randomized queue. A *randomized queue* is similar to a stack or queue, except that the item removed is chosen uniformly at random from items in the data structure. Create a generic ADT `RandomizedQueue` that supports the following operations.

```
public class RandomizedQueue<Item> {
   public RandomizedQueue()         // construct an empty randomized queue
   public boolean isEmpty()         // return true if the queue is empty, false otherwise
   public void add(Item item)       // insert the item into the queue
   public Item remove()             // delete and return an item from the queue, uniformly at random
}
```

Your ADT should support all operations in constant amortized time. That is, any sequence of N randomized queue operations (starting from an empty queue) should take O(N) steps.

Dequeue. A *double-ended queue* or *deque* (pronounced "deck") is a generalization of a stack and a queue that supports inserting and removing items from either the front or the back of the data structure. Create a generic ADT `Deque` that supports the following operations.

```
public class Deque<Item> {
   public Deque()                   // construct an empty deque
   public boolean isEmpty()         // return true if the queue is empty, false otherwise
   public void addFirst(Item item)  // insert the item at the front of the queue
   public void addLast(Item item)   // insert the item at the end of the queue
   public Item removeFirst()        // delete and return the first item in the queue
   public Item removeLast()         // delete and return the last item in the queue
}
```

Your ADT should support each operations in constant worst-case time. That is, each deque operation should take O(1) steps.

Client. Write a client program to solve each of the following problems. You may only declare one variable in your client, and it must be of type `Deque` or `RandomizedQueue`. Your client program should avoid casting by using generics.

- Given a command line parameter `k`, read in a sequence of strings from standard input, and print out a subset of exactly `k` of them, uniformly at random.
- Read in a DNA sequence from standard input using `StdIn.readChar`. Determine whether the string represents a *Watson-Crick complemented palindrome* (the string equals its reverse when you replace each base with its complement: A-T, C-G). Palindromes in DNA have have many important biological roles. For example, tumor cells frequently amplify their genes by forming DNA palindromes.

Deliverables. Submit the data types `RandomizedQueue.java` and `Deque.java`. Each data type should include its own `main` that thoroughly tests the associated operations. You may not call any Java library functions. Also submit the client programs `Subset.java` and `Palindrome.java`. Finally, submit a readme.txt file and answer the questions.