Esercizi sui TDA Deque, Array List, Node List e Sequence

1. Implementare l'interfaccia **Queue** mediante i metodi di **IndexList** (implementato dalla classe ArrayIndexList)

2. Implementare l'interfaccia **Deque** mediante i metodi di **IndexList** (implementato dalla classe ArrayIndexList). Analizzare la complessità dei metodi di **Deque** con questa implementazione.

3. Scrivere una funzione che prende in input un'istanza di **Deque** e restituisce **true** se e solo se il contenuto del **Deque** forma una sequenza palindroma.

4. Scrivere la classe **QDeque** che implementa l'interfaccia **Deque** mediante una coda. La classe **QDeque** contiene un'unica variabile di istanza di tipo **Queue**.

5. Scrivere la funzione

void removeOdd(PositionList<E> L)

che rimuove da **L** tutti gli elementi di rango dispari (il primo, il terzo, ecc.)

6. Aggiungere alla classe **NodePositionList** il metodo

PositionList <E> clone()

che restituisce una lista identica a quella su cui il metodo è invocato.

7. Scrivere la funzione

void cancellaDuplicati(PositionList<Integer> L).

La funzione prende in input un lista L di interi disposti in ordine non decrescente e cancella da L tutti i duplicati. Ad esempio se L=<2 2 3 4 4 4 6> allora dopo aver invocato cancellaDuplicati si ha L=<2 3 4 6>. Il tempo di esecuzione della funzione deve essere lineare nel numero di elementi della lista.

8. Scrivere la funzione ricorsiva

void reverse(PositionList<E> L)

che inverte la lista **L**.

9. Scrivere la funzione **merge(PositionList<Integer>L1,PositionList<Integer>L2)** che prende in input due liste ordinate **L1** ed **L2** e restituisce una lista ordinata contenente gli elementi di **L1** ed **L2**. Il tempo di esecuzione deve essere lineare nella somma delle lunghezze delle due liste.

10. Scrivere la funzione ricorsiva **mergeSort(PositionList<Integer> L)** che usa l'algoritmo merge-sort per ordinare gli elementi di **L.**

11. Scrivere la funzione

boolean isSorted(Sequence<Integer> S)

che prende in input una sequenza di interi S e restituisce true se e solo se gli elementi di S sono disposti in ordine non decrescente. La funzione deve avere tempo di esecuzione lineare nel numero di elementi della sequenza.

12. Scrivere la funzione

Position<Integer> binarySearch(Sequence<Integer>S, Integer x)

che prende in input una sequenza **ordinata** di interi S ed un intero x e restituisce la posizione di x nella sequenza. Se x non è presente nella sequenza allora la funzione restituisce null. Il tempo di esecuzione della funzione nel caso pessimo deve essere O(log n), dove n è il numero di elementi della sequenza.