LASD/SD
Anno Accademico 2009/2010
19/1/11

Cognome e Nome:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	totale
/13	22	/25	/60

1. [13 punti] Scrivere la funzione

```
public static int sum(Stack<Integer> S)
```

nella classe **ExStack_19_1_11** fornita dalla docente.

Istruzioni per lo svolgimento dell'esercizio:

- La funzione prende in input uno stack di interi e restituisce in output la somma degli interi contenuti nello stack.
- Se lo stack è vuoto la funzione deve lanciare l'eccezione EmptyStackException.
- **La funzione non deve utilizzare alcuna variabile**. **In caso contrario la funzione sarà valutata 0 punti.**

La classe di test **ExStack_19_1_11** deve essere inserita nel pacchetto in cui si trova l'interfaccia **Stack**.

2. [22 punti] Scrivere la funzione

```
public static <K,V> void insertMaxValues(Map<V,K> M, PriorityQueue<K,V> P)
```

nella classe **ExPQ_19_1_11** fornita dalla docente.

Istruzioni per lo svolgimento dell'esercizio:

- Per ciascun valore **v** delle entrate della coda a priorità **P** la funzione controlla se la mappa **M** contiene un'entrata con chiave uguale a **v** e si comporta come descritto di seguito:
  - se in origine **M** non contiene un'entrata con chiave **v** allora la funzione inserisce in **M** l'entrata (**v**,**k**) dove **k** è la chiave più grande tra quelle associate al valore **v** in **P**;
  - se in origine **M** contiene già un'entrata con chiave **v** allora la funzione non deve modificare questa entrata. Nel caso in cui non venga soddisfatto questo requisito la funzione sarà valutata al massimo **13 punti**.
- **NB: ovviamente non si devono fare assunzioni sul tipo K e, di conseguenza, sul tipo del comparatore usato per i confronti tra le chiavi di P.**

La classe di test **ExPQ_19_1_11** deve essere inserita nel pacchetto in cui si trova l'interfaccia **Priority Queue**.

3. [25 punti] Scrivere la funzione

```
public static <E> Iterable<Position<E>> visit(Tree<E> T,int k, E x)
```

nella classe **ExTree_19_1_11** fornita sul dischetto.

Istruzioni per lo svolgimento dell'esercizio:

- La funzione deve effettuare una visita ricorsiva **postorder** dell'albero e fermarsi immediatamente dopo aver incontrato **k** nodi contenenti **x**. La funzione deve restituire una collezione iterabile di tutti i nodi visitati fino a quel momento. Se **x** è contenuto in meno di **k** nodi allora la funzione deve restituire una collezione iterabile di tutti i nodi.
- Se **T** è vuoto la funzione deve lanciare l'eccezione **EmptyTreeException**.
- La funzione **NON** deve invocare funzioni che restituiscono o utilizzano collezioni/iteratori di **tutti** i nodi dell'albero. Nel caso in cui non venga soddisfatto questo requisito la funzione sarà valutata al massimo **12 punti**.

La classe di test **ExTree_19_1_11** deve essere inserita nel pacchetto in cui si trova l'interfaccia **Tree**.