Priority queue e heap

Es 1:

Scrivere una procedura per stampare i k elementi più grandi di un array utilizzando una coda di priorità. Per esempio dato un array di numeri [1, 23, 12, 9, 30, 2, 50], se viene richiesto di stampare i 3 numeri più grandi il programma deve stampare 50, 30 and 23.

Es 2:

Scrivere una funzione membro della classe PQ che restituisca il secondo valore più alto senza utilizzare la funzione getmax.

Es 3:

Aggiungere una funzione membro alla classe PQ per incrementare la priorità di un elemento:

```
void increase_key(int i, Item item)
```

se il nuovo valore di priorità non è maggiore del valore attuale di priorità dell'elemento allora la funzione deve restituire un messaggio di errore.

Es 4:

Scrivere una funzione template per determinare se gli elementi da 1 a n-1 di un array di n elementi costituiscono uno heap oppure no.

Es 5:

Scrivere un programma per simulare l'esecuzione di task della CPU di un computer aventi diversa priorità (Job-Scheduling) utilizzando una coda con priorità. Ogni task è caratterizzato da un tempo in secondi, necessario per la sua esecuzione, e da una priorità (numero intero). Simulare l'avanzamento del tempo a passi discreti di una unità. Ad ogni passo di simulazione può accadere un evento di questo tipo:

1) Il tempo rimanente al completamento del task in esecuzione (quello a priorità maggiore) viene diminuito di 1
2) Quando un lavoro è completato oppure interrotto a causa dell'arrivo di un task a priorità più elevata viene selezionato e messo in esecuzione, mediante Extract-Max, il lavoro con priorità più alta tra quelli in attesa.
3) Un nuovo lavoro può essere aggiunto con probabilità 30% in qualsiasi momento mediante una chiamata ad insert.