Programmazione II – AA 2001-02
Primo compito di Esonero

Da compilare e consegnare:

- Nome, cognome e matricola:
- Il sottoscritto, dichiara sotto la propria responsabilitá di aver superato l'esame di Programmazione I.
- Hai superato l'esame di Logica? SI [] - [] NO
- Anno di corso:

In Bocca al Lupo!

Esercizio di sbarramento. Dato un vettore `x[1 .. n]` di interi le sua *immagine speculare* é data dal vettore che contiene gli elementi in ordine inverso. Ad esempio dato il vettore

$$(1, 3, 5, 6, 7, 2, 0)$$

la sua immagine speculare é

$$(0, 2, 7, 6, 5, 3, 1).$$

Si scriva una funzione che, dati in input due vettori `x[]` ed `y[]` di interi di `n` componenti, restituisce 1 se uno é l'immagine speculare dell'altro e 0 altrimenti.

Esercizio 1. Una *rotazione* di un vettore `x[]` é il vettore che si ottiene spostando ogni elemento di un certo numero di posti "verso sinistra", assumendo che l'elemento alla sinistra di `x[0]` sia `x[n-1]`. Ad esempio, rotando di 3 posti il vettore

$$x = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$$

si ottiene il vettore

$$x' = (3, 4, 5, 6, 7, 8, 9, 0, 1, 2).$$

Si scriva una funzione C tale che, dati due vettori `x[]` ed `y[]`, restituisce 1 se uno é la rotazione dell'altro e 0 altrimenti.

Dare una stima del tempo di calcolo della vostra funzione.

Esercizio 2. Una *coda* é una struttura dati nella quale gli elementi vengono inseriti da un lato (coda) ed estratti dall'altro (testa). Implementare le funzioni di creazione della coda e di inserimento ed estrazione di un elemento. Gli elementi sono dei numeri interi.

Dare una stima della complessitá computazionale delle vostre funzioni.

Esercizio di sbarramento. Si scriva una funzione che, data una lista di interi ed un intero x, restituisce il puntatore dell'elemento che contiene x, se esso é presente, e NULL altrimenti.

Esercizio 1. Siano date due liste di interi p e q e si denoti con P e Q l'insieme degli elementi in esse contenute. Si scriva una funzione che determini se P é un sottoinsieme di Q.

Si dia una stima del tempo di calcolo della vostra funzione.

Esercizio 2. Dato un vettore `x[]` di interi, scrivere una funzione che calcoli il numero massimo di volte che un numero compare. Ad esempio, dato

$$x = (1, 0, 3, -1, -1, 0, 2, -1, 3)$$

la funzione deve restituire 3 in quanto il valore -1 é quello che compare 3 volte ed é quello che compare piú spesso.

Dare una stima della complessitá computazionale della vostra funzione.

Esercizio di sbarramento. Scrivere una funzione che verifica se un vettore di interi dato in input, di dimensione nota pari ad n, contiene un segmento iniziale della serie di Fibonacci. (La serie di Fibonacci ha i primi due elementi pari a 1 mentre ogni altro elemento é dato dalla somma dei due elementi che lo precedono: 1 1 2 3 5 8 13 ....)

Esercizio 1. Dato un vettore `x[]` di interi di n elementi, la *finestra* $[i, j]$ é il sottovettore `x[i .. j]`, cioé il sottovettore comprendente le posizioni che vanno da i a j, estremi inclusi. Il *peso* di una finestra é dato dalla somma dei suoi elementi. Ad esempio il peso della finestra $[i, j]$ é dato da $p(i, j) := x_i + x_{i+1} + = \ldots + x_j$.

Se k é compreso tra i e j, si dice che k *taglia* la finestra $[i, j]$. Ad esempio, 3 e 7 tagliano la finestra $[2, 9]$, mentre 1 e 10 no.

Si scriva una funzione che, dato un vettore di interi `x[]` di dimensione n nota, ed un valore k, calcoli il peso massimo tra tutte le finestre tagliate da k.

Dare una stima della complessitá computazionale della vostra funzione.

Esercizio 2. In questo esercizio tratteremo alberi binari i cui nodi contengono numeri interi positivi. Come di consueto diremo che un nodo x é maggiore di un nodo y se la chiave contenuta in x é maggiore di quella contenuta in y.

Un albero binario si dice *dittatoriale* se, dato un qualunque nodo x, esso é maggiore di ogni altro nodo contenuto nel suo sotto-albero (se il sotto-albero é vuoto la condizione é soddisfatta).

Si scriva una funzione che, dato un albero, verifichi se esso é dittatoriale o meno. Si puó supporre (a) che l'albero di input sia completo, cioé che ogni suo nodo eccetto le foglie abbia entrambi i figli, e (b) di poter disporre di una funzione pre-definita `isLeaf(Tree t)` che restituisce vero se `t` é una foglia e falso altrimenti.

Si dia una stima del tempo di calcolo della vostra funzione.

Esercizio di sbarramento. Scrivere una funzione che, dati in input una lista di interi ed un numero intero x, restutuisca il puntatore al primo elemento piú grande di x, se esiste, e NULL altrimenti.

Esercizio 1. Scrivere una funzione che, dati in input una lista di interi ordinata ed un valore x, inserisca x al posto giusto. Si puó assumere di avere a disposizione una funzione `myAlloc()` che restituisce il puntatore ad un elemento di tipo lista.

Dare una stima del tempo di calcolo della vostra funzione.

Esercizio 2. In questo esercizio tratteremo alberi binari i cui nodi contengono numeri interi. Come di consueto diremo che un nodo x é minore di un nodo y se la chiave contenuta in x é minore di quella contenuta in y.

Un albero binario si dice *pseudo-ordinato* se, dato un qualunque nodo x, esso é maggiore del suo figlio sinistro e minore del suo figlio destro (se il figlio manca, la condizione é soddisfatta).

Si scriva una funzione che, dato un albero, verifichi se esso é pseudo-ordinato o meno. Si puó supporre (a) che l'albero di input sia completo, cioé che ogni suo nodo eccetto le foglie abbia entrambi i figli, e (b) di poter disporre di una funzione pre-definita `isLeaf(Tree t)` che restituisce vero se `t` é una foglia e falso altrimenti.

Sia dia una stima del tempo di calcolo della vostra funzione.