02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 16/06/2015 - Prova di teoria (12 punti)

1. (1+1 punti)

Si ordini in maniera ascendente il seguente vettore di interi:

12 91 2 41 13 3 39 43 29 17 71 10 18 6 8 100

- mediante selection-sort
- mediante merge-sort.

Si indichino i passaggi rilevanti.

2. (2 punti)

Sia data una coda a priorità inizialmente vuota implementata mediante uno heap. Sia data la sequenza di interi e carattere *: 12 23 29 61 * * 18 * 34 9 13 * * 14 * 21 27 28 dove ad ogni intero corrisponde un inserimento nella coda a priorità e al carattere * un'estrazione con cancellazione del minimo. Si riporti la configurazione della coda a priorità dopo ogni operazione e la sequenza dei valori restituiti dalle estrazioni con cancellazione del minimo.

3. (2 punti)

Si determini mediante un algoritmo greedy un codice di Huffman ottimo per i seguenti caratteri con le frequenze specificate:

a: 26 b: 6 c: 4 d: 3 e: 22 f: 13 g: 14 h: 9 i: 18 l: 10

4. (1.5 punti)

Si esprima in notazione prefissa, postfissa e infissa senza parentesi la seguente espressione aritmetica mediante visita dell'albero binario corrispondente:

(((A + B) – (C – D)) * E) /((F * G) / H – I)

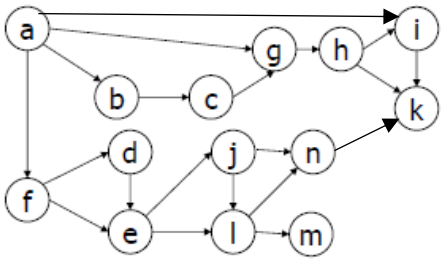
5. (1 punto)

Si supponga di aver memorizzato tutti i numeri compresi tra 1 e 500 in un albero di ricerca binario e che si stia cercando il numero 231. Quali tra queste non possono essere le sequenze esaminate durante la ricerca? Perché?

250	200	240	260	255	220	230	231				
450	100	400	150	350	200	250	231				
270	100	200	300	400	450	260	350	220	225	230	231

6.

Sia dato il seguente DAG: considerando **a** come vertice di partenza



- se ne effettui una visita in ampiezza (**1.5 punti**)
- lo si ordini in modo topologico inverso **(2 punti).**

Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 16/06/2015 - Prova di programmazione (12 punti)

1. (**4 punti**)

Si realizzi una funzione C `mul` con il seguente prototipo:

```
void mul (int *v1, int *v2, int n, int **pv);
```

che moltiplica due numeri interi di *n* cifre, il primo memorizzato nel vettore `v1`, il secondo nel vettore `v2`.
La funzione restituisce il risultato della moltiplicazione nel vettore individuato dal puntatore `pv`. I numeri sono memorizzati nei vettori `v1` e `v2` in ragione di una cifra decimale per ciascun elemento. La funzione implementa l'algoritmo classico di moltiplicazione, operando per somma e scalamento e tenendo conto dei riporti, come illustrato dal seguente esempio:

```
    0 3 2 x
    2 4 3 =
--------------
0 0 0 0 9 6 +
0 0 1 2 8  =
0 0 6 4
-------------
0 0 7 7 7 6
```

Si osservi che, per evitare overflow, il prodotto dovrà essere rappresentato su *2n* cifre. La funzione `mul` si occupi di allocare dinamicamente il vettore individuato da `pv` e di memorizzare il risultato in tale vettore.

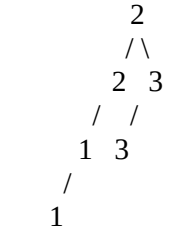
2. (**4 punti**)

Si realizzi una funzione C `doubleTree` con il seguente prototipo:

```
void doubleTree(struct node *root);
```

che, per ciascun nodo di un albero binario, crei un nodo duplicato e inserisca tale nodo come figlio sinistro del nodo originale. Ad esempio l'albero:

diventa:



Si scriva inoltre la struttura del nodo dell'albero `struct node` in grado di memorizzare chiavi intere.

3. (**4 punti**)

Un insieme di produttori-prodotti è memorizzato in una lista di liste. La lista principale specifica il nome dei produttori (stringa di al massimo 30 caratteri). Le liste secondarie riportano, per ciascun produttore, i nomi dei prodotti (stringa di al massimo 30 caratteri) e il relativo prezzo (valore reale). Tanto i nomi dei produttori quanto quelli dei prodotti sono univoci all'interno dell'intera base dati. Tanto la lista principale quanto quelle secondarie sono ordinate in ordine lessicografico crescente per nome produttore e prodotto, rispettivamente.

Si scriva una (singola) funzione ricorsiva in C in grado di ricevere quali parametri il nome di un produttore, il nome di un prodotto e gli ulteriori parametri che il candidato ritiene necessari.
Essa sia in grado di visualizzare il prezzo del prodotto del produttore indicato se esso esiste, altrimenti dia un'indicazione di errore.

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 16/06/2015 - Prova di programmazione (18 punti)

Si richiede lo sviluppo di un'applicazione in grado di gestire un impianto sciistico secondo le seguenti modalità:

- ogni sciatore ha una tessera caratterizzata da un identificatore (`cardId`) costituito da un numero intero
- ogni skilift ha un identificatore di skilift (`skiliftId`) costituito da una stringa di 10 caratteri. Ogni skilift ha un lettore di tessere per abilitare uno sciatore all'utilizzo dello skilift stesso.

Il sistema riceve da tastiera le letture degli skilift con formato:

```
skiliftId cardId time
```

verifica e rilascia o meno l'autorizzazione al passaggio dello sciatore. Il tempo viene introdotto quale valore intero rappresentante il numero di minuti trascorsi a partire dalle ore `00:00` del giorno stesso.

L'obbiettivo è di evitare che persone diverse usino la stessa tessera. L'autorizzazione viene quindi data solo se la tessera non è stata letta dallo stesso skilift per un certo intervallo di tempo. Tale intervallo di tempo dipende dalla posizione e lunghezza dello skilift.

I tempi dai vari skilift vengono letti da un file. Il nome del file viene ricevuto dall'applicazione come parametro sulla linea di comando e ha il seguente formato:

```
skiliftId   timeInterval
```

dove `timeInterval` è un valore espresso in minuti.

Il sistema, una volta inizializzato, deve:

- fornire una funzione di autorizzazione del tipo

  ```
  int authorize (long cardId, char *skiliftId, int time);
  ```

  che permetta allo sciatore `cardId` di utilizzare lo skilift `skiliftId` oppure gli neghi l'accesso se il vincolo temporale, valutato in base all'ultimo passaggio dello sciatore su tale skilift e l'ora attuale `time`, non è rispettato
- mantenere, in memoria centrale, l'elenco di tutti gli sciatori abilitati da ciascuno skilift e, per ciascuno skilift, il numero di volte per cui lo sciatore è stato abilitato
- mantenere, in memoria centrale, l'elenco di tutti gli skilift utilizzati da ciascuno sciatore e per ciascuno skilift l'ora dell'ultimo utilizzo (abilitazione).

Si osservi che:

- essendo il numero di sciatori relativamente elevato e variabile durante la giornata, il costo asintotico di tutte le operazioni effettuate deve essere al più logaritmico nel numero di sciatori
- essendo il numero di skilift limitato e deducibile dalla lettura del file riportante i ritardi temporali, il costo asintotico di tutte le operazioni non ha alcun vincolo in relazione al numero di skilift.

PER ENTRAMBE LE PROVE DI PROGRAMMAZIONE (18 o 12 punti):

- ***indicare nell'elaborato e nella relazione (oltre a nome, cognome e numero di matricola) anche il nome del corso per cui si sta sostenendo l'esame (AP, APA I+II)***
- ***è consentito utilizzare chiamate a funzioni <u>standard</u>, quali ordinamento per vettori, funzioni su FIFO, LIFO, liste, BST, tabelle di hash, grafi e altre strutture dati, considerate come librerie esterne. Gli header file delle librerie utilizzate devono essere allegati all'elaborato. Le funzioni richiamate, inoltre, dovranno essere incluse nella versione del programma allegata alla relazione. I modelli delle funzioni ricorsive non sono considerati funzioni standard***
- ***consegna delle relazioni (per entrambe le tipologie di prova di programmazione): entro venerdì 19/06/2015, alle ore 24:00, mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). QUALORA IL CODICE SPEDITO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. Si ricorda che la valutazione del compito viene fatta senza discussione o esame orale, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***