02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II
Appello del 14/06/2016 - Prova di teoria (12 punti)

1. (1 punto)
Sia data la seguente sequenza di coppie, dove la relazione i-j indica che il nodo i è adiacente al nodo j:

1-9, 2-3, 0-5, 4-2, 0-8, 3-6, 0-10, 1-5, 6-9, 9-4

si applichi un algoritmo di on-line connectivity con quickunion, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale. I nodi sono denominati con interi tra 0 e 10.

2. (0.5 + 0.5 punti)
Si ordini in maniera ascendente il seguente vettore di interi:

10 23 6 78 54 6 90 20

mediante merge-sort (**0.5 punti**) e bottom-up merge-sort (**0.5 punti**). Si indichino le strutture dati usate nei passi intermedi.

3. (2 punti)
Sia dato un albero binario con 13 nodi. Nella visita si ottengono le 3 seguenti sequenze di chiavi:

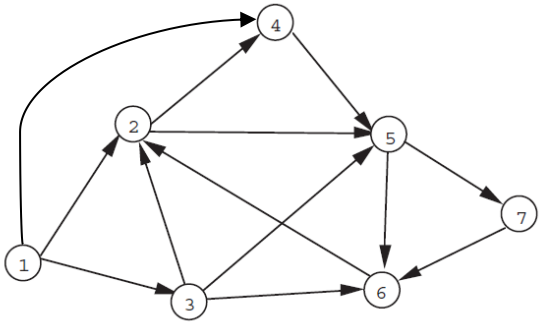
preorder	10	5	16	7	20	17	4	2	1	7	8	13	11
inorder	16	7	17	20	4	5	10	1	2	8	13	7	11
postorder	17	4	20	7	16	5	1	13	8	11	7	2	10

Si disegni l'albero binario di partenza.

4. (1 punto)
Si supponga di aver memorizzato tutti i numeri compresi tra 1 e 300 in un BST 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
150	300	200	250	220	240	235	231
270	100	200	300	260	190	220	231

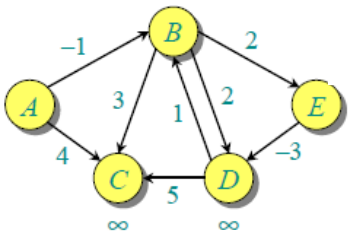
5. (2.0 + 1.5 + 1.5 punti)
Sia dato il seguente grafo orientato:



- se ne effettui una visita in profondità, considerando **1** come vertice di partenza (**2 punti**)
- lo si ridisegni, etichettando ogni suo arco come T (tree), B (back), F (forward), C (cross), considerando **1** come vertice di partenza (**1.5 punti**)
- se ne effettui una visita in ampiezza, considerando **1** come vertice di partenza (**1.5 punti**).

Qualora necessario, si trattino i vertici secondo l'ordine alfabetico.

6. (1.5 + 0.5 punti)
Considerando **A** come vertice di partenza, si determinino mediante l'algoritmo di Bellman-Ford i valori di tutti i cammini minimi che collegano **A** con ogni altro vertice (**1.5 punti**). Si riportino i passaggi rilevanti. L'algoritmo di Dijkstra sarebbe pervenuto allo stesso risultato? Si giustifichi la risposta (**0.5 punti**).



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

Appello del 14/06/2016 - Prova di programmazione (18 punti)

Un negozio online ha la necessità di gestire il proprio catalogo di prodotti contenuto in un file con il seguente formato:

productID productName productPrice #availability category

dove:

- productID è il codice univoco alfanumerico di 10 caratteri
- productName è il nome (stringa di dimensione variabile di massimo 255 caratteri)
- productPrice è il prezzo (valore reale positivo)
- #availability è il numero di unità disponibili (valore intero non negativo)
- Category: è un codice univoco alfanumerico di 10 caratteri.

Per quanto riguarda le categorie si noti che ogni prodotto appartiene a una specifica categoria merceologica. Le categorie sono al massimo 100. Non si acquisisce in modo esplicito un elenco di codici di categoria. L'elenco va costruito durante l'inserimento dei dati

Il catalogo deve poter permettere le seguenti operazioni:

1. inserimento di un nuovo prodotto o aggiunta di unità disponibili per un prodotto già esistente
2. ricerca di un prodotto nel catalogo dato il codice
3. ricerca di un prodotto dato codice e categoria. Occorre localizzare prima la categoria e limitare quindi la ricerca ai prodotti della categoria
4. stampa dei prodotti di una data categoria ordinati per codice o nome a scelta dell'utente
5. ricerca di un prodotto per nome: il nome può essere parziale (dato solo dalla parte iniziale terminata col carattere asterisco '*'): in tal caso è possibile che la ricerca dia più risultati
6. valutazione di soddisfacibilità di un ordine.

Per i punti 2. e 3. la ricerca per prodotti deve essere al peggio di costo logaritmico nel numero di prodotti. Per il punto 3. la ricerca per categorie deve essere al peggio di costo lineare nel numero di categorie. Per il punto 5. la ricerca deve ritornare TUTTI gli elementi che rispettino i criteri di ricerca, ossia tutti i prodotti col medesimo nome. Per il punto 6. il programma deve acquisire da file un elenco di prodotti desiderati in quantità indicate e valutare se questi siano disponibili. Sia tale file caratterizzato dal seguente formato:

1. sulla prima riga è presente un unico intero N ad indicare il numero di prodotti d'interesse
2. sulle N righe successive sono presenti N coppie <codice_prodotto> <quantità>, in ragione di una per riga, a indicare i contenuti dell'ordine.

In caso l'ordine sia soddisfacibile, si provveda anche a calcolare l'ammontare complessivo dell'ordine stesso e ad aggiornare le disponibilità dei prodotti in magazzino.

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).***
- ***Se non indicato diversamente, 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 devono essere allegati all'elaborato (il loro contenuto riportato nell'elaborato stesso). 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 17/06/2016 ore 24:00 mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). QUALORA IL CODICE CARICATO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. Si ricorda che la valutazione del compito viene fatta senza discussione o esame orale, sulla base dell'elaborato svolto in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

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

Appello del 14/06/2016 - Prova di programmazione (12 punti)

1. (2 punti)

Scrivere una funzione

```
void matMax (int **m, int r, int c);
```

che riceve una matrice m di interi di r righe e c colonne e visualizza (a video) la posizione di tutti gli elementi che sono strettamente maggiori di tutti gli elementi adiacenti. Ad esempio, data la matrice seguente:

	0	1	2	3
0	5	2	3	1
1	4	1	6	4
2	3	0	5	2

occorre visualizzare: (0, 0) e (1, 2).

2. (4 punti)

Si scriva una funzione `separaParole` che, a partire da una stringa contenente parole separate da spazi, generi un vettore di stringhe (vettore di puntatori a caratteri) contenente le singole parole, prive di spazi e duplicate mediante allocazione dinamica. La funzione deve essere richiamabile, ad esempio, nel modo seguente:

```
char frase[1000], **parole;
int n, i;
…
fgets(frase,1000,stdin);
n = separaParole(frase,&parole);
for (i=0; i<n; i++)
  printf("parola %d -> %s\n", parole[i]);
…
```

Si scriva il prototipo e il contenuto della funzione. Si noti che il puntatore parole viene utilizzato per un vettore (allocato dinamicamente) di puntatori a `char`: ogni casella punterà a una delle parole generate dalla stringa originale. Il vettore di stringhe viene ritornato per riferimento (o meglio, se ne passa per valore il puntatore). Per semplicità è lecito ipotizzare che gli spazi che separano le parole non siano multipli.

3. (6 punti)

Sia U l'insieme degli interi compresi 1 e 8: U= {1, 2, 3, 4, 5, 6, 7, 8}. Sia S una matrice n x 9 di interi che su ciascuna delle n righe contiene un sottoinsieme di U terminato dal valore 0. La matrice viene ricevuta da una funzione come parametro formale con numero di righe ignoto. Il numero di righe compare come ulteriore parametro. Dato un intero k, ulteriore parametro della funzione, si scriva una funzione ricorsiva in C che visualizzi, se esiste, una collezione di k sottoinsiemi la cui unione sia U. La funzione abbia il seguente prototipo:

```
void cover(int S[][9], int n, int k);
```

Esempio: