

## Esame del 24 marzo 2003

### Parte I (teoria)

Tempo: 45 minuti, NON è possibile consultare libri o appunti.

#### Esercizio 1 (3 punti)

Si consideri il seguente vettore:

3 5 9 4 2 8 7 6 3 1 5

Si assuma di volerlo ordinare usando l'algoritmo noto come *Counting Sort*. Tale algoritmo utilizza nel caso considerato un vettore di appoggio C di dimensione pari a 9. Si riportino i valori assunti dagli elementi del vettore C dopo l'esecuzione del primo e del secondo ciclo di calcoli.

#### Esercizio 2 (3 punti)

Si consideri un albero binario di ricerca inizialmente vuoto. Su di esso vengono eseguite in sequenza le operazioni di inserimento di elementi con le seguenti chiavi:

40 18 34 56 21 89 86 32 10 50 44

Si disegnino

- la struttura dell'albero risultante al termine degli inserimenti indicati
- la struttura dell'albero dopo la cancellazione dell'elemento con chiave 18.

### Esercizio 3 (3 punti)

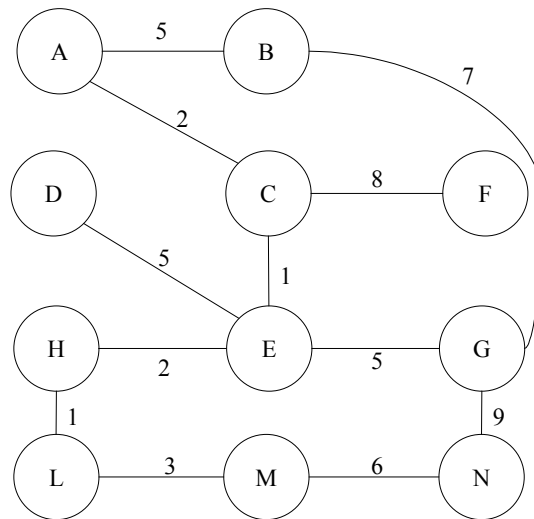
Si consideri una tabella di hash basata su open addressing che utilizza la tecnica del linear probing per risolvere le collisioni. La tabella è inizialmente vuota. Su di essa viene eseguita la seguente sequenza di operazioni di inserimento di elementi con chiave corrispondente ad una stringa:

SCANO  
TORINO  
MILANO  
GENOVA  
OLBIA  
SAVONA  
CAGLIARI

Si mostri il contenuto della tabella di hash al termine della sequenza di inserimenti, assumendo che la sua dimensione sia pari a  $M=9$  e che la funzione di hash utilizzata corrisponda per ciascuna chiave alla somma modulo  $M$  dei numeri d'ordine dei caratteri che compongono la stringa all'interno dell'alfabeto italiano ( $A=0, B=1, C=2, \dots$ ).

### Esercizio 4

Sia dato il seguente grafo:



#### Punto a (3 punti)

Si disegni la matrice di adiacenza e la lista di adiacenza del grafo, e si determini la dimensione in byte di ciascuna delle due strutture.

#### Punto b (3 punti)

Si determini l'albero di copertura minimo prodotto utilizzando l'algoritmo di Kruskal, indicando il peso totale di tale albero. Qualora necessario, si trattino i vertici e gli archi secondo l'ordine alfabetico.

## Parte II (programmazione)

**Tempo: 60 minuti, è possibile consultare libri o appunti.**

Si progetti un programma C per analizzare le relazioni tra un gruppo di amici. Il programma legge le informazioni a partire da due file di testo:

- Il file **persone.txt** vi sono, uno per riga, in un ordine arbitrario, i nomi delle persone del gruppo (senza spazi interni). Sulla prima riga del file vi è un numero che indica il numero dei nomi presenti.
- Il file **amici.txt** elenca (una per riga) le *coppie* di persone che hanno una relazione di amicizia tra di loro. Ogni coppia è identificata dai nomi delle due persone, separati da uno spazio. Si tenga conto che la relazione di amicizia è simmetrica, per cui ogni coppia di persone compare una sola volta. Sulla prima riga del file vi è un numero che indica il numero di coppie presenti.

Il programma permette all'utente di elencare gli *amici diretti* e *indiretti* di ciascuna persona: quando l'utente immette un nome di una persona, il programma stampa (in ordine alfabetico) i nomi di tutte le persone che hanno relazioni di amicizia (direttamente o indirettamente) con essa.

È richiesta la definizione della struttura dati che si intende utilizzare, i prototipi di tutte le principali funzioni ed una loro descrizione.

### Esempio

<b>persone.txt</b>	<b>amici.txt</b>
6	4
Angelo	Angelo Bruno
Bruno	Carlo Dario
Carlo	Dario Enrico
Dario	Fabio Angelo
Enrico	
Fabio	

Persona? Bruno

Gli amici di Bruno sono:

Angelo Fabio

N.B. il testo introdotto dall'utente compare sottolineato.