Informatica Generale II - Prova teorica

A.A. 20052006

Esame: 25 gennaio 2007
Codice: YKQB

1. Si consideri il seguente codice:

```
struct data {
   int giorno;
   int mese;
   int anno;
   data(int g, int m, int a){ giorno=g; mese=m; anno=a;}
};
```

   La seguente dichiarazione:

```
  data oggi;
```

   (a) produce un errore a tempo d'esecuzione. Infatti, la suddetta dichiarazione risulta nell'invocazione del costruttore standard senza argomenti che però è stato inibito automaticamente dal compilatore a fronte della dichiarazione del costruttore a tre argomenti;
   (b) produce un errore a tempo di compilazione. Infatti, la suddetta dichiarazione risulta nell'invocazione del costruttore standard senza argomenti che però è stato inibito automaticamente dal compilatore a fronte della dichiarazione del costruttore a tre argomenti;
   (c) produce un errore a tempo di compilazione in quanto l'allocazione di una variabile avente come tipo una struttura definita dal programmatore può essere allocata solo dinamicamente tramite l'operatore `new`;
   (d) è corretta
   (e) non rispondo

2. Si analizzi il seguente codice:

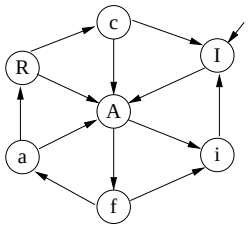
```
void foo(int A[], int N) {
  for (int i=0; i < N-1; ++i) {
    int min = i;
    for (int j=i+1; j < N; ++j)
      if (A[j] < A[min]) min = j;
    swap(A[i], A[min]);
  }
}
```

   Esso implementa l'algoritmo:

   (a) Bubblesort
   (b) Quicksort
   (c) Mergesort
   (d) Selection sort
   (e) non rispondo

3. Dato il grafo e la sua rappresentazione con matrice delle adiacenze riportati in figura 1:

   (a) La visita in ampiezza dal vertice I produce la seguente sequenza d'uscita: I c i A f R a
   (b) La visita in ampiezza dal vertice I produce la seguente sequenza d'uscita: I A f i a R c



	c	R	a	f	i	I	A
c	0	0	0	0	0	1	1
R	1	0	0	0	0	0	1
a	0	1	0	0	0	0	1
f	0	0	1	0	1	0	0
i	0	0	0	0	0	1	0
I	0	0	0	0	0	0	1
A	0	0	0	1	1	0	0

Figura 1:

(c) Nessuna delle risposte è accettabile;
(d) La visita in ampiezza dal vertice I produce la seguente sequenza d'uscita: I A f a R c i
(e) non rispondo

4. Si supponga di avere una funzione `visit` di visita di un albero che inserisce il valore del nodo visitato in fondo ad una lista. Per ottenere una lista ordinata in senso crescente dall' attraversamento di un BST utilizzando la funzione di visita `visit`:

   (a) la domanda per com'è posta non ha senso
   (b) dipende dai valori presenti nel BST
   (c) bisogna attraversare il BST in preorder
   (d) bisogna attraversare il BST inorder
   (e) non rispondo

5. Si consideri la rappresentazione di un grafo mediante *lista delle adiacenze* (sia n il numero di nodi e m il numero di archi). Quale tra le seguenti affermazioni è *falsa*?

   (a) l'operazione di accesso a tutti i successori di un nodo può essere effettuata in modo più efficiente che non se fosse stata usata la rappresentazione mediante matrice delle adiacenze;
   (b) la verifica dell'esistenza di un arco da un nodo i a un nodo j si può effettuare in modo meno efficiente che non se fosse stata usata la rappresentazione mediante matrice delle adiacenze;
   (c) tale rappresentazione richiede un'occupazione di memoria proporzionale a $(n + m)$;
   (d) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero di nodi del grafo;
   (e) non rispondo

6. Le operazioni di base di stack e code sono:

   (a) push e pop per stack, put e get per code
   (b) push e pop per code, put e get per stack
   (c) push e pop
   (d) put e get
   (e) non rispondo

7. Si consideri la seguente funzione:

```
int* funz(int dim) {
  int* punt;
  punt=new int[dim];
  for (int i=0; i < dim; ++i) punt[i]=i;
  return punt;
}
```

   La memoria allocata all'interno della funzione `funz` mediante l'operatore `new`:

   (a) viene deallocata quando il controllo passa all'esterno della funzione `funz`
   (b) non può più venire deallocata poiché l'operatore `delete` non compare all'interno della funzione stessa

(c) rimane allocata fin quando il programma non termina o finché non viene esplicitamente deallocata tramite l'istruzione `delete`
(d) rimane allocata fin quando il programma non termina o finché non venga esplicitamente deallocata tramite una istruzione `delete []`
(e) non rispondo

8. Quale delle seguenti sequenze di complessità asintotiche rappresenta un ordinamento crescente dal più piccolo al più grande?

(a) $O(\log n)$ $O(n)$ $O(n^3)$ $O(n^n)$ $O(n^2 \log n)$ $O(2^n)$ $O(n^{\log n})$
(b) $O(n)$ $O(n^3)$ $O(n^n)$ $O(\log n)$ $O(n^2 \log n)$ $O(2^n)$ $O(n^{\log n})$
(c) $O(\log n)$ $O(n)$ $O(n^2 \log n)$ $O(n^3)$ $O(n^n)$ $O(2^n)$ $O(n^{\log n})$
(d) $O(\log n)$ $O(n)$ $O(n^2 \log n)$ $O(n^3)$ $O(2^n)$ $O(n^{\log n})$ $O(n^n)$
(e) non rispondo

9. In quale ordine partendo dalla radice vengono visitati i nodi dell'albero in figura 2 da un algoritmo di attraversamento pre-order?

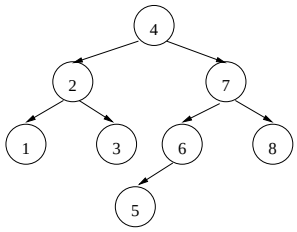


Figura 2: Albero

(a) 12345678
(b) 85673124
(c) 87654321
(d) 42137658
(e) non rispondo

10. La lunghezza del cammino di un albero è

(a) la somma dei livelli di tutti i nodi del percorso piu' lungo dalla radice dell'albero ad una foglia.
(b) l'altezza massima del percorso dalla radice ad una foglia.
(c) la somma dei livelli di tutti i nodi dell'albero.
(d) l'altezza minima del percorso dalla radice ad una foglia.
(e) non rispondo