Informatica Generale II - Prova teorica

A.A. 20052006

Esame: 25 gennaio 2007
Codice: SGJY

1. Supponendo di aver definito una funzione `funzEsame` con la seguente definizione:

```
void funzEsame (double i, double & j) {
  j += i;
}
```

   L'invocazione *scorretta* della suddetta funzione `funzEsame` avrà la forma:

   (a)
```
double i=1.0, j=5.0;
funzEsame(i, j+1);
```
   (b)
```
double i=1.0, j=5.0;
funzEsame(5.0, j);
```
   (c)
```
double i=1.0, j=5.0;
funzEsame(i+1, j);
```
   (d)
```
double i=1.0, j=5.0;
funzEsame(i, j);
```
   (e) non rispondo

2. Si consideri il seguente frammento di codice:

```
struct Tipo1 {
 Tipo2 a;
};

struct Tipo2 {
 Tipo1 a;
};
```

   (a) è errato perché Tipo2 non ha un costruttore di default.
   (b) è corretto, ma Tipo2 non è istanziabile.
   (c) è errato perché Tipo1 e Tipo2 contengono due campi con lo stesso nome.
   (d) è errato perché Tipo1 e Tipo2 non sono istanziabili.
   (e) non rispondo

3. 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 venga esplicitamente deallocata tramite una istruzione `delete []`

(d) rimane allocata fin quando il programma non termina o finché non viene esplicitamente deallocata tramite l'istruzione `delete`

(e) non rispondo

4. Si supponga di avere la sequenza di numeri *3,8,6,4,5*. Dopo *3* iterazioni soltanto di Bubblesort su tale sequenza, il risultato sarà:

   (a) 3,4,8,5,6
   (b) 3,4,8,6,5
   (c) 3,8,4,5,6
   (d) 3,4,5,8,6
   (e) non rispondo

5. In quale ordine partendo dal nodo 1 vengono visitati i nodi del grafo in figura 1 da un algoritmo di visita in ampiezza?

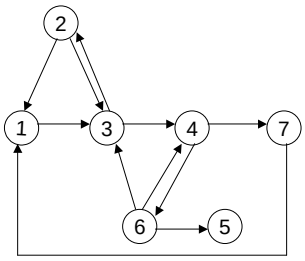

Figura 1:

   (a) Nessuna risposta è accettabile;
   (b) 1346572
   (c) 1342756
   (d) 1342675
   (e) non rispondo

6. Quale è la lunghezza del cammino dell'albero in figura 2

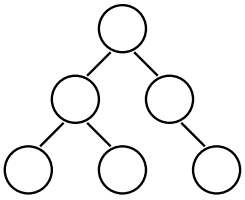

Figura 2:

   (a) 6
   (b) 8
   (c) 9
   (d) 2
   (e) non rispondo

7. Si consideri l'algoritmo di fusione (merge) tra due array ordinati aventi n elementi ciascuno:

   (a) si implementa sempre tramite programmazione ricorsiva;

(b) ha complessità $\mathcal{O}(n^2)$;
(c) si avvale di questo algoritmo l'ordinamento per fusione che funziona nel seguente modo: si sceglie a caso un elemento di pivot sulla base del cui valore si divide in due parti l'array, di seguito ciascuna parte viene ricorsivamente ordinata e infine i due array vengono fusi;
(d) ha complessità lineare poiché ogni volta che si inserisce un elemento nell'array risultato si esegue un numero costante di operazioni;
(e) non rispondo

8. La seguente funzione `foo`:

```
void foo(Node * x, Node * y) {
  y->next = x->next;
  x->next = y;
}
```

(a) inserisce il nodo y tra il nodo x e il successore di x
(b) inserisce la lista puntata da y dopo il nodo x
(c) concatena due liste concatenate x e y
(d) inserisce la lista puntata da x dopo il nodo y
(e) non rispondo

9. Si consideri un grafo: sia n il numero di nodi e m il numero di archi. Quale tra le seguenti affermazioni è *falsa*?

(a) l'algoritmo BFS richiede tempo proporzionale a $(n + m)$
(b) l'algoritmo DFS richiede tempo proporzionale a $(n + m)$;
(c) l'algoritmo BFS si ottiene dall'algoritmo DFS iterativo sostituendo la Coda con una Stack;
(d) l'algoritmo BFS si ottiene dall'algoritmo DFS iterativo sostituendo lo Stack con una Coda;
(e) non rispondo

10. Quale tipo di attraversamento d'albero implementa il seguente codice?

```
void foo-order(Node* l, void visit(Node*))
  StackPtr s = new Stack();

  Push(s, i);
  while(! StackIsEmpty(s)) {
    Node * h = Pop(s);

    visit(h);
    if (h->right != NULL) Push(s, h->right);
    if (h->left != NULL) Push(s, h->left);
  }
  delete s;
}
```

(a) level-order
(b) inorder
(c) postorder
(d) preorder
(e) non rispondo