Prova scritta di Algoritmi e Strutture Dati (1^o anno)

Febbraio 2001

L'ordine in cui vengono svolti gli esercizi non è rilevante.

I punti previsti per ogni esercizio si riferiscono ad uno svolgimento completamente corretto.

NOME:

COGNOME:

MATRICOLA:

==

Non scrivere qui sotto

Esercizio	Punti previsti	Punti assegnati
1	4	
2	3	
3	7	
4	7	
5	4	
6	5	
Totale	30	

Esercizio 1 (punti 4)

Consideriamo il seguente codice C:

```
#include <stdio.h>

typedef struct elem {
  int i;
  struct elem *s;
} elem;

typedef elem *le;

void aaa(le *ee, int val)
{
  le x;
  x = (le)malloc(sizeof(elem));
  x->i = val;
  x->s = *ee;
  *ee = x;
}

void bbb(le lll)
{ int t;
  t = 0;
  while (lll != NULL)
    { t = t + lll->i;
      printf("%d\n",t);
      lll = lll->s;
    }
}

main()  {  int i; le l;
           l = NULL;
           for (i=10;i>0;i--) aaa(&l,i);
           bbb(l);
          }
```

Domande:

1. Descrivere qual'è lo scopo della procedura aaa (indipendentemente dal main)
2. Descrivere qual'è lo scopo della procedura aaa (indipendentemente dal main)
3. Dire qual'è l'output del programma

Esercizio 2 (punti 3)

Consideriamo il seguente codice C:

```
#include <stdio.h>

int f(int *x, int *y)
{ x = y;
  *x = 50;
  return(*y);
}

main()
{ int a = 100;
  int b = 10;
  int c;
  c = f(&a, &b);
  printf("%d, %d, %d\n", a, b, c);
}
```

Domanda: qual'è l'output del programma?

Esercizio 3 (punti 7)

Consideriamo il tipo di dato **Successione** di interi implementato mediante liste con puntatori.

Domande:

1. Definire in pseudo-codice (o in C) i tipi utilizzati per implementare le successioni.
2. Definire in pseudo-codice (o in C) la primitiva Empty che crea una successione vuota.
3. Definire in pseudo-codice (o in C) la primitiva Insert-t che aggiunge un elemento in testa alla lista.
4. Definire in pseudo-codice (o in C) l'implementazione dell'operazione seguente:

   $$\text{Concatena} : \text{Succ} \times \text{Succ} \longrightarrow \text{Succ}$$

   tale che Concatena(s,t) prende due liste s e t e restituisce la lista ottenuta concatenando t in coda a s.

   Le successioni in input sono modificate dall'operazione, non se ne fa una copia; non bisogna allocare nuove celle ma riutilizzare quelle esistenti in s e t; l'operazione deve funzionare anche se una o entrambe le liste in input sono vuote.

Esercizio 4 (punti 7)

Consideriamo il tipo di dato *Insieme* di numeri interi nell'intervallo da 0 a K-1, implementato con una tabella hash **aperta** con B bucket.

Domande:

1. Definire in pseudo-codice (o in C) i tipi utilizzati per l'implementazione del tipo di dato;
2. Definire una funzione di hashing per indirizzare la tabella nel caso K=1.000.000 e B=500;
3. Definire in pseudo-codice (o in C) la funzione che inizializza una tabella vuota;
4. Definire in pseudo-codice (o in C) la primitiva Delete che elimina un elemento dalla tabella

   N.B.: se nell'algoritmo si utilizzano altre primitive sul tipo di dato bisogna definire anche quelle in pseudo-codice (o in C); cercare di cancellare un elemento che non è in tabella NON è un errore e in questo caso l'operazione non ha nessun effetto sulla tabella stessa.

Esercizio 5 (punti 4)

Consideriamo il seguente pezzo di programma:

```
i, j : integer;  r: float;
r ← 1.0;
per i = 1,...,n :
{
   j ← 1;
   while j ≤ i do
   {
      r ← r * 2;
      j ← j + 1
   }
}
scrivi(r)
}
```

Domanda: determinare la complessità della procedura in Θ funzione di n.

Non fare conti troppo dettagliati esplicitando tutte le costanti, ma considerare il costo di tutte le parti di programma mettendo eventualmente in evidenza le operazioni dominanti (una volta che si sono individuate si possono ignorare le altre). Non limitarsi a dare solo il risultato: bisogna spiegare come ci si arriva.

Esercizio 6 (punti 5)

Consideriamo la seguente procedura ricorsiva in C.

```
int fff(int a[]; int inf; int sup)
{
  int k; int m;
  if (inf>=sup)
     return(a[inf]);
  else
      { m = (sup * inf)/ 2;
        if (a[inf]>a[sup])
             k = fff(a,m,sup)
        else
             k = fff(a,inf,m);
        return(k);
      }
}
```

Domanda: Stimare la complessità in $\Theta()$ della chiamata `fff(A,0,N-1)` in funzione di `N`.

NOTE:

- I passaggi di parametro si intendono secondo le regole del C;
- Non è necessario fare conti molto dettagliati esplicitando tutte le costanti. Basta considerare il costo di tutte le parti di programma mettendo eventualmente in evidenza le operazioni dominanti (una volta che si sono individuate si possono ignorare le altre).
- Non limitarsi a dare solo il risultato: bisogna spiegare come ci si arriva.