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

Settembre '99

NOTE:

- L'ordine in cui vengono svolti gli esercizi non è rilevante (in altre parole: se avete problemi a svolgerne uno potete passare tranquillamente ai successivi e farlo dopo).
- I punti previsti per ogni esercizio si riferiscono ad uno svolgimento completamente corretto.
- Nei calcoli di complessità (esercizi 4, 5, 6) non fare conti troppo dettagliati esplicitando tutte le costanti, ma non limitarsi neppure a dare solo il risultato: bisogna spiegare come ci si arriva.

NOME:

COGNOME:

MATRICOLA:

==========

Non scrivere qui sotto

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

Esercizio 1 (punti 4) Consideriamo il seguente codice C:

```
#include <stdio.h>

typedef struct n {  int i;
                    struct n *s;
                 } n;
typedef n *l;

void p(l *a, int num)
{ l a1;
  a1 = (l)malloc(sizeof(n));
  a1->i = num;
  a1->s = *a;
  *a = a1;
}

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

main()  {  l aa;
           aa = NULL;
           p(&aa,1000);
           p(&aa,100);
           p(&aa,10);
           pr(aa);
         }
```

Domande:

1. Descrivere in generale cosa fa la funzione p
2. Descrivere in generale cosa fa la funzione pr
3. Dire qual'è l'output del programma

Esercizio 2 (punti 3)

Consideriamo il seguente codice C:

```
#include <stdio.h>

int a = 100;

void ppp (int x, int *y, int *z);

int fff (int x, int y);

main()
{
  int a = 10;
  int b = 5;
  int c;
  a = fff(a,b);
  ppp( a, &b, &c);
  printf ("%d, %d, %d\n", a, b, c);
}

void ppp (int x, int * y, int *z)
{ int a, b;
  a = x + (*y);
  *y = a - 2;
  b = x - 5;
  *z = (a + b) / 10;
}

int fff (int x, int y)
{
  return( (x + y)*a );
}
```

Domanda: qual'è l'output?

Esercizio 3 (punti 6)

Consideriamo il tipo di dato **Successione** di interi implementato mediante array.

Consideriamo l'operazione

$$\text{Purge} : \text{Succ} \times \text{Int} \longrightarrow \text{Succ}$$

Definita (a parole) come segue:

Purge(s,x) elimina da s tutte le occorrenze di x (la successione s viene modificata dall'operazione, non se ne fa una copia).

Domande:

1. Definire in pseudo-codice i tipi utilizzati per implementare le successioni.
2. Definire in pseudo-codice l'implementazione di Purge come procedura.

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=100.000 e B=1.000;
3. Definire in pseudo-codice (o in C) la funzione che inizializza una tabella vuota;
4. Definire in pseudo-codice (o in C) un algoritmo che stampa il contenuto della tabella in ordine crescente (N.B.: se nell'algoritmo si utilizzano primitive sul tipo di dato oppure una funzione di ordinamento, bisogna definire anche quelle in pseudo-codice);
5. Valutare la complessità dell'algoritmo, in $\Theta()$, in funzione del numero B di bucket, del numero n di interi presenti in tabella, del numero K di possibili valori e dire se esiste un caso peggiore (N.B.: (1) secondo l'implementazione scelta, non è detto che tutti e tre i parametri siano rilevanti, bisogna capire quali lo sono; (2) se si è implementata la Member a parte, bisogna valutare la complessità di quella e utilizzarla nella valutazione di complessità dell'algoritmo).

Esercizio 5 (punti 5)

Consieriamo la procedura seguente in pseudo-codice:

```
procedure ppp(aa : array[1..n,1..n] of integer);
{
   i,j,k: integer;
   per i = 1,...,n :
     per j = 1,...,n :
       per k = 1,...,j-1 :
          a[i][j] = a[i][j] + a[i][k];
}
```

Domanda: determinare la complessità della procedura in $\Theta()$ in funzione di n.

Esercizio 7 (punti 5+3)

Consideriamo la seguente procedura ricorsiva:

```
procedure r(aa : array[1..n] of integer IN-OUT;
                 inf, sup: integer IN)
{
   bb : array[1..n] of integer;
   i,j : integer;
   if (sup>inf) then
   {
      med = (inf+sup) div 2;
      r(aa, inf, med);
      r(aa, med+1, sup);
      j = 1;
      per i = med+1,..,sup:
        {  bb[j] = aa[i];
           j++;
        }
      per i = inf,..,med:
        {  bb[j] = aa[i];
           j++;
        }
      per i = inf,...,sup :
         aa[i] = bb[i];
   }
}
```

Domanda: stimare la complessità della procedura in $\Theta()$ in funzione di n nell'ipotesi che n sia una potenza di 2 (N.B.: il parametro aa si intende passato per riferimento).

Domanda jolly (3 punti): dire in generale qual'è l'effetto di una chiamata `r(aa,1,n)` su un generico array aa.