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

Giugno 2000

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.

NOME:

COGNOME:

MATRICOLA:

==

Non scrivere qui sotto

Esercizio	Punti previsti	Punti assegnati
1	3	
2	4	
3	12	
4	6	
5	5	
Totale	30	

Esercizio 1 (punti 3)

Consideriamo il seguente codice C:

```
#include <stdio.h>

int a = 10;

int fff (int x);

void ppp (int * x);

main()
{
  int a = 4;
  int b = 2;
  b = fff(a);
  ppp(&b);
  printf ("%d, %d\n", a, b);
}

void ppp (int * x)
{
  *x = fff(*x);
}

int fff (int x)
{ x = x + a;
  return( x - 5 );
}
```

Domanda: Dire qual'è l'output del programma

Esercizio 2 (punti 4)

Consideriamo il seguente codice C:

```
#include <stdio.h>

float A[3][3];
float d2(float X[2][2]);
void  extra(float M3[3][3], float M2[2][2], int r0, int r1, int c0, int c1);
float d3(float X[3][3]);

main()
{
   int i,j;
   for ( i=0 ; i<3 ; i++ )
      for ( j=0 ; j<3 ; j++)
         scanf("%f",&A[i][j]);

   printf("%f\n",d3(A));
}

float d2(float X[2][2])
{  return(X[0][0]*X[1][1]-X[0][1]*X[1][0]);
}

void extra(float M3[3][3], float M2[2][2], int r0, int r1, int c0, int c1)
{ M2[0][0]=M3[r0][c0]; M2[0][1]=M3[r0][c1];
  M2[1][0]=M3[r1][c0]; M2[1][1]=M3[r1][c1];
}

float d3(float X[3][3])
{  float m0,m1,m2;
   float B[2][2];
   extra(X,B,1,2,1,2);
   m0=d2(B);
   extra(X,B,1,2,0,2);
   m1=d2(B);
   extra(X,B,1,2,0,1);
   m2=d2(B);
   return(X[0][0]*m0-X[0][1]*m1+X[0][2]*m2);
}
```

Domande:

1. Dire qual'è l'operazione calcolata dalla funzione d2.

2. Dire qual'è l'operazione calcolata dalla funzione d3.
3. Dire qual'è l'output per un input 3 4 2 6 1 -1 -2 3 0

Esercizio 3 (punti 12)

Consideriamo il tipo di dato *Dizionario* di numeri interi (senza duplicati) implementato con alberi binari di ricerca (con puntatori).

Domande:

1. Definire in pseudo-codice i tipi utilizzati per l'implementazione del tipo di dato;
2. Definire in pseudo-codice la primitiva **Member**;
3. Definire in pseudo-codice la primitiva **Print_Dict** che prende in input il dizionario e ne stampa il contenuto in ordine crescente;
4. Definire in pseudo-codice la primitiva

   Range_Member : Dizionario $\times$ Int $\times$ Int $\longrightarrow$ Lista-di-interi

   dove **Range_member**(D,min,Max) restituisce la lista di tutti i valori nell'intervallo [min,Max], estremi compresi, che si trovano nel dizionario D.

   - Il tipo Lista-di-interi in output deve essere implementato con puntatori: definire i tipi necessari.
   - Se nessuno dei valori dell'intervallo è nel dizionario viene restituita una lista vuota (non è un errore!).
   - Se min > Max si interpreta come un intervallo vuoto (anche questo non è un errore!).
   - TUTTI i valori interi sono ammissibili: non si può usare nessun valore speciale.

Esercizio 4 (punti 6)

Consideriamo la funzione seguente in pseudo-codice:

```
procedure qqq(aa : array[1..n] of integer IN) : integer;
{
 i, j, t : integer;
 b : array[1..n] of integer
 per i = 1,...,n :
    b[i] ← aa[i];
 per i = 1,...,n :
 {
    j ← n;
    while j > i do
    {
       b[i] ← b[i] + a[j];
       j−−
    }
 }
 t ← 0;
 per i = 1,...,n :
    t ← t + b[i];
 return(t);
}
```

Domanda: determinare la complessità della procedura in funzione della dimensione n del parametro.

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 5 (punti 5)

Consideriamo la seguente procedura ricorsiva che lavora su alberi binari di interi implementati con puntatori (campo info per il valore del nodo, campi sx e dx per i puntatori ai figli).

```
procedure v(t : albero binario);
{
  if (t non vuoto) then
      { scrivi(t->info);
        if t->info > 0 then
            v(t->sx)
        else
            v(t->dx);
      }
}
```

Domanda: Stimare la complessità della procedura in $\Theta()$ in funzione del numero di nodi n dell'albero, nel caso peggiore (dire qual'è il caso peggiore).