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

Giugno 2001

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	18	
4	5	
Totale	30	

Esercizio 1 (punti 3)

Consideriamo il seguente codice C:

```
#include <stdio.h>
#define  MAX 128

char S[MAX];

void r(char *ss, int n);

main()  {  int i,len;
           char c;
           for (len=0; len < MAX; len++)
           { scanf( "%c", &c );
             if (((c>='a')&&(c<='z'))||(c==' ')) S[len]=c;
             else {S[len]='\0'; break;}
           }

           r(S,len);

           for ( i = 0; S[i]!='\0'; i++ ) printf( "%c", S[ i ] );
           printf("\n");
        }

void r(char *ss, int n)
{ int m, i; char c;
     m = n / 2;
     for ( i = 0; i < m; i++ )
     {
        c = ss[ i ];
        ss[ i ] = ss[ n-i-1 ];
        ss[ n-i-1 ] = c;
     }
}
```

Domande:

1. Dire qual'è l'effetto della funzione r sulla stringa che riceve come **parametro attuale**.
2. Dire qual'è l'output del programma avendo in input la stringa seguente:
   elicaf oizicrese nu! ottircs eneiv non otseuq am

Esercizio 2 (punti 4)

Consideriamo il seguente codice C che lavora su liste dinamiche:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct cc {  int ii;
                     struct cc *nn;
                  } cc;
typedef cc *ll;

void mkl(ll *L, int x);
void in1(ll L, int x);
int mmb(ll L, int x);
void prt(ll L);

main()  {  ll L; int n,i;
           L = NULL;
           for (i=0;i<10;i++)
           {  scanf("%d",&n);
              if (L==NULL) mkl(&L,n);
              else in1(L,n);
           }
           printf("? "); scanf("%d",&n);
           printf("%d\n", mmb(L,n));

           prt(L);
        }

void mkl(ll *L, int x)
{
   *L = (ll)malloc(sizeof(struct cc));
   (*L)->ii = x; (*L)->nn = NULL;
}

void in1(ll L, int x)
{
   while (L->nn!=NULL) L = L->nn;
   L->nn = (ll)malloc(sizeof(struct cc));
   L->nn->ii = x;
   L->nn->nn = NULL;
}
```

```
int mmb(ll L, int x)
{
   while (L!=NULL)
     if (L->ii==x) return(1);
     else L=L->nn;
   return(0);
}

void prt(ll L)
{
  for (;L!=NULL;L=L->nn) printf("%d, ",L->ii);
  printf("\n");
}
```

Domande:

1. Dire a quale primitiva del tipo di dato *successione* assomiglia la funzione `in1` e in che cosa differisce dall'implementazione standard di tale primitiva.

2. Dire qual'è l'effetto della funzione `mmb` sulla lista che riceve come **parametro attuale**.

3. Dire qual'è l'output del programma per il seguente input:

   1 2 3 4 5 6 7 8 9 0
   2

Esercizio 3 (punti 18)

Consideriamo il tipo di dato *Insieme* di *Stringhe di caratteri* implementato come segue:

- le strighe sono implementate mediante array di caratteri di una dimensione fissata da una costante di programma MAX;
- il carattere nullo $'\backslash 0'$ viene usato come "tappo" per segnalare la prima casella dell'array che non appartiene alla stringa, nel caso la stringa sia più corta di MAX;
- sono valide tutte le stringhe di lunghezza tra 0 e MAX (quindi anche la stringa vuota!) composte di caratteri diversi dal carattere nullo;
- le stringhe si confrontano tra loro secondo l'ordine lessicografico (quello dell'elenco telefonico): secondo questo ordine la stringa vuota precede tutte le altre;
- gli insiemi sono implementati con alberi binari di ricerca (con puntatori).

Domande:

1. Definire in pseudo-codice (o in C) le costanti ed i tipi necessari per l'implementazione del tipo di dato;
2. Definire in pseudo-codice (o in C) l'operazione **StrConf** che confronta due stringhe e dice qual'è la minore;
3. Definire in pseudo-codice (o in C) la primitiva **Insert** che inserisce una stringa nel dizionario;
4. Definire in pseudo-codice (o in C) la primitiva **Delete** che cancella una stringa dal dizionario;
5. Definire in pseudo-codice (o in C) l'operazione **ContaChar** che conta il numero totale di caratteri che compongono le stringhe presenti nel dizionario.
6. Valutare la complessità dell'operazione **ContaChar** implementata al punto precedente in $\Theta()$ in funzione del numero n di stringhe presenti nel dizionario, del numero totale m di caratteri presenti in tali stringhe e della dimensione massima MAX di una stringa.
   N.B.: secondo l'implementazione fatta non necessariamente tutti i parametri n, m e MAX sono necessari per esprimere tale complessità.

Esercizio 4 (punti 5)

Consideriamo la seguente funzione ricorsiva in C:

```
function s(int a[]; int x; int inf; int sup)
{ int med,r,i;
  if (inf>sup) return(0);
  else if (inf==sup) return a[inf];
  else { r = 0;
         for (i=inf;i<=sup;i++) r+=a[i];
         med = (inf+sup)/2;
         if (x<a[med]) return( r + s(a,x,inf,med) );
         else return( r + s(a,x,med+1,sup) );
       }
}
```

Domanda: Stimare la complessità della chiamata `s(aa,xx,0,n-1)` in $\Theta()$ in funzione di `n`.