**Fondamenti di Informatica L-A (A.A. 2008/2009) - Ingegneria Informatica**
**Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Lunedì 22 Dicembre 2008 – durata 2h**
**Compito A**
**Compito Modalità Fondamenti di Informatica L-A (a.a. precedenti)**

**ESERCIZIO 1 (10 punti)**

Una banca vuole memorizzare in un array di strutture dati di tipo **`titolo`** le informazioni relative ad alcuni titoli azionari. In ogni struttura di tipo **`titolo`** viene memorizzato il codice unico identificativo del titolo azionario (un intero detto **`codTitolo`**) e i valori di apertura, di chiusura e il valore medio di tale titolo azionario (tutti e tre i valori rappresentati tramite il tipo float**`)`**. Il problema sorge dal fatto che ogni sera la Borsa comunica alla banca due array, il primo, di interi, contenente i codici dei titoli (ogni codice compare due sole volte in tale array); il secondo, nelle posizioni corrispondenti al primo vettore, contiene i valori di apertura e chiusura dei titoli (il primo valore incontrato è sempre quello riferito all'apertura, mentre il secondo valore incontrato è sempre riferito alla chiusura). Per ogni titolo, infatti, la Borsa comunica il valore di apertura ed il valore di chiusura, da cui due valori differenti per ogni titolo, e la ripetizione del codice di ogni singolo titolo due volte. Si tenga conto inoltre che la borsa memorizza i codici dei titoli in ordine casuale, e quindi il valore di apertura ed il valore di chiusura dello stesso titolo spesso non sono in posizioni consecutive (ma il valore di apertura di un titolo precede sempre il valore di chiusura dello stesso titolo).

a) Il candidato realizzi una funzione:

```
titolo * valMedio(int codTitoli[], float valori[], int dim, int * resultDim)
```

che riceva come parametri un vettore di interi rappresentate i codici dei titoli **`codTitoli`**, un vettore di float, detto **`valori`**, rappresentante i valori di ogni titolo (il valore di un titolo è memorizzato nella posizione corrispondente al codice del titolo nel primo vettore), e la loro dimensione **`dim`**; questi due array sono i dati che la Borsa ogni sera comunica alla banca. La funzione deve determinare i valori di apertura e chiusura, e calcolare il valore medio di ogni titolo. A tal scopo, la funzione deve restituire in un apposito vettore di strutture di tipo **`titolo`**, allocato dinamicamente (non necessariamente della dimensione strettamente necessaria), il codice di un titolo, valori di apertura, chiusura e valor medio (senza titoli ripetuti). Tramite il parametro **`resultDim`**, passato per riferimento, la funzione restituisce anche la dimensione logica dell'array di strutture **`titolo`**. Si riporti, per completezza, anche la definizione della struttura dati **`titolo`**.

b) La borsa italiana comunica i dati tramite due file di testo **`codici.txt`** e **`valori.txt`**. Nel primo su ogni riga è presente un codice titolo, mentre nel secondo è presente nella riga corrispondente il valore di apertura o di chiusura. Il candidato realizzi un semplice **`main`**, in cui vengono letti dai file e memorizzati in due array i codici ed i valori relativi ai titoli (i file contengono al massimo informazioni riguardo 100 titoli distinti, quindi vi saranno al più 200 righe), e si usi la funzione di cui al punto a) con tali dati per generare un vettore di strutture **`titolo`** e stamparne a video le coppie codice titolo/valore medio.

**ESERCIZIO 2 (10 punti)**

Si supponga di avere a disposizione, già definito, l'ADT lista per interi (denominato **`list`**, con relative primitive). Il candidato definisca una funzione iterativa:

```
list ripetuti(list l1);
```

La funzione deve restituire una nuova lista contenente i valori di **`l1`** che compaiono ripetuti più volte (cioè almeno due volte), eventualmente non in maniera ordinata. La nuova lista non deve contenere elementi ripetuti. La funzione deve essere implementata accedendo alle liste tramite la notazione a puntatore, e senza fare ricorso alle primitive dell'ADT.

Al fine di semplificare la funzione di cui sopra, il candidato realizzi una funzione *ricorsiva*:

```
int conta(int el, list l1);
```

che conta quante volte l'elemento **el** compare in **l1**. Tale funzione deve essere implementata utilizzando le sole primitive dell'ADT lista.

Ad esempio, se **l1 = {1, 3, 2, 4, 2, 1, 3, 1, 1, 2}**, la funzione restituirà una lista **{1, 3, 2}**, poiché il valore 1 compare 4 volte, il valore 3 compare 2 volte ed il valore 2 compare 3 volte.

## ESERCIZIO 3 (6 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
int dim = 4;

char * gambero(char v[], int dim, int step){
    int i;
    char * result = (char*) malloc(sizeof(char) * (dim+1));

    for(i=0; i<dim; i++)
        result[(i+step)%dim] = v[i%dim];
      result[dim] = '\0';
    return result;
}

int main(){
      char msg[] = "Bologna";
      char * temp = msg;
      int dim=0;
      for ( ; *temp!='\0'; ) {
            temp+=1;;
            dim+=1;
      }
      temp = gambero(msg, dim, 3);
      printf("%s %d", temp, dim);
      return (0);
}
```

## ESERCIZIO 4 (3 punti)

Si consideri la grammatica G con scopo S, simboli non terminali {V, W, X, Y, Z} e simboli terminali {a, b, c, d, e, 1, 2, 3, 4}:

```
S::= VZ | WZ
V::= WXV | WYZ
W::= XYX | ZYZ
X::= a | b | c
Y::= d | e
Z::= 1 |2 | 3 | 4
```

La stringa "aebcadce23" appartiene al linguaggio generato da tale grammatica?
In caso affermativo, se ne mostri la derivazione left-most.

## ESERCIZIO 5 (3 punti)

Il candidato introduca brevemente il concetto di compilatore e di interprete di programmi, mostrandone le differenze.

**ESERCIZIO 1**

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

typedef struct {
      int codTitolo;
      float ap;
      float ch;
      float valore;
} titolo;

titolo * valMedio(int codTitoli[], float valori[], int dim, int * resultDim) {
      int i, j;
      titolo * result;
      int presente;

      result = (titolo *) malloc((dim/2) * sizeof(titolo));
      *resultDim = 0;

      for (i=0; i<dim; i++) {
            presente = 0;
            for (j=0; j<*resultDim && !presente; j++)
                  if (codTitoli[i] == result[j].codTitolo)
                        presente = 1;
            if (!presente) {
                  for (j=i+1; j<dim && (codTitoli[i] != codTitoli[j]); j++);
                  result[*resultDim].codTitolo = codTitoli[i];
                  result[*resultDim].ap = valori[i];
                  result[*resultDim].ch = valori[j];
                  result[*resultDim].valore = (valori[i] + valori[j])/2;
                  *resultDim = *resultDim + 1;
            }
      }
      return result;
}

int main(void) {
      titolo * media;
      int mediaDim;
      int i;
      int codTitoli[200];
      float valori[200];
      FILE * fp1, *fp2;
      int dim1 = 0, dim2 = 0;

      fp1 = fopen("codici.txt", "r");
      fp2 = fopen("valori.txt", "r");
      if (fp1==NULL || fp2==NULL) exit(-1);

      while (fscanf(fp1, "%d", &(codTitoli[dim1])) ==1) dim1++;
      while (fscanf(fp2, "%f", &(valori[dim2])) ==1) dim2++;
      media = valMedio(codTitoli, valori, dim1, &mediaDim);
      for (i=0; i< mediaDim; i++)
            printf("%d %f %f %f\n",
                  media[i].codTitolo, media[i].ap, media[i].ch, media[i].valore);
      fclose(fp1); fclose(fp2);
      return (0);
}
```

**ESERCIZIO 2**

```
int conta(int el, list l1) {
      if (empty(l1))
            return 0;
      else
            if (head(l1) == el)
                  return 1 + conta(el, tail(l1));
            else
                  return conta(el, tail(l1));
}

list ripetuti(list l1) {
      list result = NULL;
      list temp;
      int el;

      while (l1 != NULL) {
            el = l1->value;
            if (conta(el, l1) > 1 && conta(el, result)==0 ) {
                  temp = (item*) malloc(sizeof(item));
                  temp ->value = el;
                  temp ->next = result;
                  result = temp;
            }
            l1 = l1->next;
      }
      return result;
}
```

**ESERCIZIO 3**

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
**gnaBolo 7**

La funzione `main()` dapprima calcola la lunghezza della stringa `msg`, e poi invoca la funzione `gambero(…)`.
Tale funzione alloca spazio sufficiente per contenere la stringa passata tramite il parametro `v`, più uno spazio ulteriore per il terminatore di stringa. Il ciclo `for` successivo provvede a copiare i caratteri di `v` nello spazio di memoria appena allocato, ma operando uno "shift" a destra di `step` posizioni dei caratteri, avendo cura di settare comunque il terminatore di stringa.
La funzione `main()` infine stampa a video la stringa ottenuta dalla funzione `gambero(…)`, e la lunghezza della stringa.

**ESERCIZIO 4**

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S -> VZ -> WXVZ -> XYXXVZ -> aYXXVZ -> aeXXVZ -> aebXVZ -> aebcVZ -> aebcWYZZ -> aebcXYXYZZ -> aebcaYXYZZ -> aebcadXYZZ -> aebcadcYZZ -> aebcadceZZ -> aebcadce2Z -> aebcadce23