Fondamenti di Informatica T-1 (A.A. 2013/2014) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 20 Febbraio 2014 – durata 1h
Totale 12 punti, sufficienza con 7

Compito B

ESERCIZIO 1 (6 punti)

Dati un array di interi `s1` e una lista di interi `l2` con lo stesso numero di elementi, si realizzi una funzione ITERATIVA

```
list change(int *s1, list l2, int length);
```

che prenda in ingresso, oltre al vettore e alla lista, la loro lunghezza `length` e che restituisca una nuova lista di interi di dimensione `length` contenente un elemento dell'array e un elemento della lista alternati: se `length` è dispari il primo elemento da inserire deve appartenere a `s1`, altrimenti a `l2`. Notare bene che, per ogni posizione, deve essere inserito o un elemento di `s1` o un elemento di `l2`, non entrambi; l'elemento non inserito deve quindi essere ignorato.

Per esempio, se `s1 = {1,3,7,5,8}`, `l2 = [4,2,8,0,9]` e `length = 5`, la funzione `change()` deve restituire la lista `[1,2,7,0,8]`. Infatti, il primo elemento inserito nella lista da restituire è l'intero 1 appartenente all'array, in quanto `length` è dispari. Il secondo intero della lista restituita invece corrisponde al secondo elemento della lista `l2` (l'intero 2), mentre il numero 7 è incluso nella lista restituita in quanto terzo elemento dell'array (si salta quindi l'intero 3).

La funzione `change()` dovrà essere implementata utilizzando le sole primitive dell'ADT lista; ogni altra funzione dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `change()` creata.

Nota: l'ordine degli elementi della lista restituita dalla funzione `change()` è ininfluente.

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):

+ 62 – 99

ESERCIZIO 3 (3 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>
#include <stdlib.h>
#include <string.h>

#define N 5
#define SEP '.'

int myConverter(char *str){
      int i, res = 0;

      for (i = 0; str[i] != '\0'; i++)
            res = res*10 + ( str[i] - '0' );

      return res;
}

int fun(char *s){
      int k;
      char *a;
      int l = strlen(s);

      a = s;

      for(k=0; k<l; k++){
            if(*a == SEP){
                  return 0;
            }
            a++;
      }
      return myConverter(s);
}

int main(void){
      int res=0, i=0;
      char *v[] = {"75", "8.8", "3", "17.5", "12"};

      while(i < N){
            res = res + fun(v[i]);
            i++;
      }

      printf("%d\n",res);
      return (0);
}
```

ESERCIZIO 4 (1 punto)

Si descriva brevemente cos'è un ADT in linguaggio C e come si può costruire fornendo semplici esempi di codice.

Soluzioni

ESERCIZIO 1

```
list change(int* s1, list l2, int length){
      list res = emptylist();

      while(length>0){
            if(length % 2 == 1)
                  res = cons(s1[0], res);
            else
                  res = cons(head(l2), res);
            l2 = tail(l2);
            s1++;
            length--;
      }
      return res;
}

int main(){
      list l2, res;
      int length = 5;
      int s1[] = {1,3,7,5,8};

      l2 = cons(4, cons(2, cons(8, cons(0, cons(9, emptylist())))));

      printf("change\n");
      res = change(s1,l2,length);
      while( ! empty(res) ){
            printf("%d", head(res));
            res = tail(res);
      }

      return 0;
}
```

ESERCIZIO 2

+62 - 99 = - 37

+ 99 → 01100011 = 64 + 32 + 2 + 1
10011100
- 99 → 10011101
+ 62 → 00111110 = 32 + 16 + 8 + 4 + 2
- 37 → 11011011 = -128 + 64 + 16 + 8 + 2 + 1

ESERCIZIO 3

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

```
90
```

La funzione `main()` crea un array `v` di puntatori a stringhe ed esegue un `while` che, ad ogni ciclo, invoca la funzione `fun()` passando come parametro di ingresso la i-esima stringa contenuta nell'array `v`. Il numero intero restituito dalla funzione `fun()` è poi sommato alla variabile `res` che ad ogni iterazione mantiene il risultato della somma parziale.

La funzione iterativa `fun()` controlla inizialmente se la stringa `s` contiene il carattere di separazione `SEP`: se sì, la funzione termina restituendo il valore intero `0`, altrimenti invoca la funzione `myConverter()` passando come parametro di ingresso la stringa ben formata `s`. Il risultato della funzione invocata è poi restituito al `main()`.

La funzione iterativa `myConverter()` ha il compito di convertire una stringa in numero intero. Per fare questo, la funzione analizza ogni singolo carattere della stringa d'ingresso che rappresenta una cifra numerica, partendo dalla prima posizione. Visto che la conversione da `char` a `int` considera il corrispondente valore ascii del carattere, è necessario sottrarre a tale valore quello del carattere `'0'` in modo da ottenere esattamente il valore decimale desiderato. Inoltre, ad ogni ciclo il risultato parziale è moltiplicato per 10 per riflettere il valore posizionale degli interi decimali.

La funzione `main()` stampa sullo standard output il valore della variabile `res` che rappresenta la somma dei soli valori interi derivati dalla conversione delle stringhe presenti all'interno del vettore `v`, ovvero "75", "3" e "12".