Fondamenti di Informatica T-1 (A.A. 2012/2013) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 11 Luglio 2013 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Data una lista di stringhe ben formate `listStr` e un puntatore a carattere `word`, si realizzi una funzione RICORSIVA

```
list onlyIfStartsWith(list listStr, char* word);
```

che restituisca una nuova lista contenente gli elementi della lista `listStr` che iniziano con la stringa ben formata `word`. Ad esempio se `listStr = [ "Ciao", "caio", "caldo", "arCa". "mare" ]` e `word = "ca"`, la funzione `onlyIfStartsWith()` deve restituire la lista `[ "caio", "caldo" ]`, ovvero i soli valori della lista `listStr` che iniziano con la stringa `"ca"`.

A tal fine si implementi e usi opportunamente la funzione

```
int startsWith(char* str, char* word);
```

che restituisce 1 se la stringa `str` inizia con la stringa `word`, 0 altrimenti.

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

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

ESERCIZIO 2 (2 punti)

Si consideri la grammatica G con scopo S e simboli terminali {f, g, h, x, y, z}

```
S ::= A B | B A
A ::= P B Q | P Q
B ::= Q A P | Q P
P ::= x | y | z
Q ::= f | g | h
```

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

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>

int* fun(int *v, int a, int b){
      int i, *t2 = v, *t1, *res;
      while(*v>=0) v++;
      res = t1 =(int*)malloc(sizeof(int)*(v-t2));
      for(i=0;i<(v-t2);i++){
            if( t2[i] > a && t2[i] < b ){
                  *t1 = *(t2+i);
                  t1++;
            }
      }
      *t1 = -1;
      return res;
}

int main(){
      int *res;
      int a[]= {3, 5, 0, 4, 2, -1, 6, 2, 8, 9, 0};
      res = fun(a, 2.9, 5);
      while(*res > 0){
            printf("%d", *res);
            res++;
      }
      return 0;
}
```

ESERCIZIO 4 (1 punti)

Si descriva brevemente in che cosa consiste la differenza tra una funzione ricorsiva e una funzione iterativa, sottolineando le differenze principali in relazione all'uso della memoria dell'elaboratore.

Soluzioni

ESERCIZIO 1

```
int startsWith(char* str, char* word){
      while( *word != '\0' ){
            if( *word != *str){
                  return 0;
            }
            word++;
            str++;
      }
      return 1;
}

list onlyIfStartsWith(list l, char *word){
      if( empty(l) ){
            return l;
      }
      else {
            if( startsWith(head(l), word) ){
                  return cons( head(l), onlyIfStartsWith(tail(l),word) );
            }
            else {
                  return onlyIfStartsWith(tail(l),word);
            }
      }
}

int main(){
      list l, res;

      l = cons("Ciao", cons("caio", cons("caldo", cons("arCa",
                                                cons("mare", emptylist())))));

      printf("onlyIfStartsWith\n");
      res = onlyIfStartsWith(l,"ca");
      while( ! empty(res) ){
            printf("%s\n", head(res));
            res = tail(res);
      }
      printf("\n");

      return 0;
}
```

ESERCIZIO 2

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S → BA → QPA → gPA → gyA → gyPQ → gyzQ → gyzf

ESERCIZIO 3

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

```
34
```

La funzione `main()` invoca la funzione `fun()` passando come parametri di ingresso un array di interi e i valori 2.9 e 5.

La funzione iterativa `fun()` itera lungo l'array `v` fino al primo valore negativo escluso e poi alloca dinamicamente spazio di memoria sufficiente a contenere tali elementi. In seguito la funzione `fun()` inserisce nell'area di memoria allocata dinamicamente gli elementi di `v` il cui valore è compreso tra 2 e 5 (notare che il valore 2.9 è stato troncato a 2). Infine la funzione `fun()` inserisce `-1` come ultimo elemento dell'area di memoria allocata dinamicamente e restituisce un riferimento al primo elemento di tale area.

La funzione `main()` stampa sullo standard output i valori presenti nell'area di memoria allocata dinamicamente.