

Fondamenti di Informatica T-1 (A.A. 2018/2019) - Ingegneria Informatica
Prof.ssa Mello
Prova Scritta – 14 Febbraio 2019 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Si scriva una funzione RICORSIVA `selectRec`

```
list selectRec(list l, int n)
```

che, data una lista di numeri interi `l`, dia in uscita una lista composta dai soli numeri pari se $n < 0$, dai soli numeri dispari se $n \geq 0$.

Ad esempio se `l = [8, 9, 4, 2, 1]` e `n = -3`, la funzione `selectRec()` deve restituire la lista `[8, 4, 2]`; se `l = [8, 9, 4, 2, 1]` e `n = 2`, la funzione `selectRec()` deve restituire la lista `[9, 1]`.

Si realizzi, poi, una funzione ITERATIVA `findMax`

```
list findMax(list l)
```

che data in ingresso la lista prodotta da `selectRec` restituisca **una lista** contenente il valore massimo. Se la lista è vuota restituirà la lista vuota.

Le funzioni dovranno essere implementate utilizzando le sole primitive dell'ADT lista, includendo "list.h"; ogni altra funzione dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che crei la lista `l`, invochi correttamente le funzioni `selectRec()` con `n = -3` e `findMax()` e stampi in output il risultato.

ESERCIZIO 2 (2 punti)

Si consideri la seguente grammatica G con scopo S , simboli non terminali $\{A, B\}$ e simboli terminali $\{1, 2, x, y, +, -, (,)\}$:

```
S ::= A | B  
A ::= (C+C) | C+A | C+B  
B ::= (E-C) | E-B | E-A  
C ::= 1 | 2  
E ::= x | y
```

La stringa " **$x-1+y-(2+1)$** " appartiene al linguaggio di 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? (Riportare il risultato stampato, **ben evidenziato**, e motivare opportunamente la risposta data).

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

void K(float* v, int d){
    int i;

    for (i = 0; i < d; i++){
        *(v+i) = - (v[i]);
    }
    return;
}

void X(float* vec, int dim, float up, float down, int num[]){
    int i;

    if(up){
        K(vec, dim);

        for(i=3; i>0; --i){
            if(vec[dim-i] < up && *(vec+dim-i) > down)
                *(num+i-1) = (int)vec[dim - i];
            else if(vec[dim-i] > up)
                *(num+i-1) = (int)up;
            else
                *(num+i-1) = (int)down;
        }
    }

    return;
}

int main(){
    int i;
    float prices[] = {7.9, -4.3, -10.8};
    int v[3];

    X(prices, 3, 6.8, -3.7, v);

    for(i = 0; i < 3; ++i){
        printf("%d", v[i]);
    }
}
```

ESERCIZIO 4 (1 punto)

Si illustri il concetto di allocazione dinamica di un blocco di memoria (`malloc`) e se ne fornisca un breve esempio.

Soluzioni

ESERCIZIO 1

```
#include <stdio.h>
#include "list.h"

// RICORSIVA
list selectRec(list l, int n){
    if(empty(l)){
        return emptyList();
    }else{
        if(n < 0 && (head(l)%2)==0){
            return cons(head(l), selectRec(tail(l), n));
        }
        else if(n >= 0 && (head(l)%2)==1){
            return cons(head(l), selectRec(tail(l), n));
        }
        else
            return selectRec(tail(l), n);
    }
}

list findMax(list l){
    int max;

    if(empty(l)){
        return emptyList();
    }

    max = head(l);
    l = tail(l);
    while(!empty(l)){
        if(head(l) > max)
            max = head(l);
        l = tail(l);
    }
    return cons(max, emptyList());
}

int main(){
    list l, selected, res;
    l = cons(8, cons(9, cons(4, cons(2, cons(1, emptyList()))));

    selected = selectRec(l, -3);
    res = findMax(selected);
    if(!empty(res)){
        printf("MAX: %d\n", head(res));
    }

    return 0;
}
```

ESERCIZIO 2

$S \rightarrow B \rightarrow E-A \rightarrow x-A \rightarrow x-C+B \rightarrow x-1+B \rightarrow x-1+E-A \rightarrow x-1+y-A \rightarrow x-1+y-(C+C)$
 $\rightarrow x-1+y-(2+C) \rightarrow x-1+y-(2+1)$

ESERCIZIO 3

Il programma è corretto e l'output prodotto è:

6
4
-3

La funzione `main` inizializza un array `prices` con i valori $\{7.9, -4.3, -10.8\}$ e invoca la funzione `X` passando come parametri `prices`, l'intero `3`, due float `'6.8'` e `'-3.7'` ed un array di 3 elementi `v`.

La funzione `X` riceve in input un puntatore a float `vec`, un intero `dim`, due float `up` e `down` ed un array di interi `num`. `vec` diventa quindi puntatore all'array `prices`, mentre `dim` assume il valore `3`, `up` il valore `6.8`, `down` il valore `-3.7` e `num` diventa un puntatore al primo elemento dell'array `v`. All'interno della funzione `X` viene creata una variabile intera `i`. La prima condizione logica è `true` (`up > 0`). Si entra quindi nell'`if` dove viene invocata la funzione `K` passando come parametri `vec` e `dim`.

La funzione `K` riceve come input un puntatore a float `v` ed un intero `d`. `v` diventa quindi puntatore all'array `prices` (in quanto `vec` a sua volta è un puntatore all'array `prices`) e `dim` assume il valore `3`. All'interno della funzione `K` vengono invertiti di segno tutti i valori dell'array di cui `v` è puntatore. All'inizio del ciclo `for`, infatti, `i` assume il valore `0`; il ciclo si ripete finché `i` è minore di `3`. Ad ogni iterazione del `for`, il valore contenuto nella cella di memoria `i` posizioni più avanti a quella riferita dal puntatore `v` viene sostituito con il valore contenuto nella posizione `i`-esima dell'array di cui `v` è puntatore invertito di segno.

Dopo `K`, `prices` è: $\{-7.9, 4.3, 10.8\}$

Una volta eseguita la funzione `K`, la funzione `X` esegue un ciclo `for`. All'inizio, `i` assume il valore `3`; il ciclo si ripete finché `i` è maggiore di `0`. Ad ogni iterazione, se il valore contenuto nella posizione `dim-i` del vettore puntato da `vec` è compreso fra le due soglie `up` e `down`, allora viene inserito nell'array di interi `num` in posizione `i-1` (il valore sarà troncato, casting a `int`), altrimenti se è maggiore del valore della soglia `up` viene inserito il valore di quest'ultima (anch'esso troncato, casting a `int`), altrimenti viene inserito il valore della soglia `down` (anch'esso troncato, casting a `int`). Il vettore puntato da `vec` viene quindi analizzato dal primo all'ultimo elemento; i valori vengono inseriti nell'array `num` partendo dall'ultima posizione alla prima.

Dopo `X`, `num` è: $\{6, 4, -3\}$

Infine, la funzione `main` stampa il contenuto dell'array `v`.

Dopodiché il programma termina.