Fondamenti di Informatica T-1 (A.A. 2010/2011) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Mercoledì 13 Luglio 2011 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Date due liste di interi `l1` e `l2` (liste non vuote), si realizzi una funzione ITERATIVA

```
list selection(list l1, list l2);
```

che restituisca una nuova lista contenente gli elementi di `l1` che compaiono in `l2` un numero di volte almeno uguale al loro valore.

Ad esempio se `l1 = [1, 3, 2, 4]` e `l2 = [3, 5, 2, 3, 1, 2, 2]`, la funzione `selection()` deve restituire la lista `[1, 2]`, ovvero i soli valori della lista `l1` che rispettano il requisito dato. Infatti l'intero 3 compare solo due volte nella lista `l2`, mentre l'elemento 4 nessuna volta e quindi né 3 né 4 vengono inseriti nella lista restituita.

La funzione `selection()` 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 `selection()` creata.

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

ESERCIZIO 2 (2 punti)

Si consideri la grammatica G con scopo S e simboli terminali {1, 2, 3, p, q, r}

```
S ::= A X | A Y
X ::= X B | B
Y ::= Y A X | A
A ::= 1 | 2 | 3
B ::= p | q | r
```

La stringa "123pq" 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>
#include <string.h>

char* merge(char* charsA, int dimA, char* charsB){
	char *res;
	int dimRes, count=0, i;

	while( charsB[count] != '\0' ){
		count++;
	}
	dimRes = dimA + count + 1;
	res = (char*) malloc(sizeof(char)*dimRes);

	for( i = 0; i < dimRes-1; i++ ){
		if( *charsB=='\0' ){
			res[i] = *charsA;
			charsA++;
		}
		else{
			res[i] = *charsB;
			charsB++;
		}
	}
	res[i] = '\0';

	return res;
}

int main() {
	char charsA[] = {'c', 'h'};
	char charsB[] = "afg";
	char *res;
	res = merge(charsA, 2, charsB);
	printf("%s\n", res);
	return 0;
}
```

ESERCIZIO 4 (1 punti)

Il candidato illustri brevemente la differenza tra passaggio per valore e passaggio per riferimento.

Soluzioni

ESERCIZIO 1

```
list selection(list l1, list l2){
      int count, el;
      list temp;
      list res = emptylist();
      while( ! empty(l1) ){
            temp = l2;
            count = 0;
            el = head(l1);
            while( ! empty(temp) ){
                  if( head(temp) == el ) count++;
                  temp = tail(temp);
            }
            if( count >= el ){
                  res = cons(el, res);
            }
            l1 = tail(l1);
      }
      return res;
}

int main(){
      list l1, l2, res;
      l1 = cons(1, cons(3, cons(2, cons(4, emptylist()))));
      l2 = cons(3, cons(5, cons(2, cons(3, cons(1, cons(2, cons(2,
                                                      emptylist()))))))));
      res = selection(l1, l2);
      while( ! empty(res) ){
            printf("%d\n", head(res));
            res = tail(res);
      }
      return 0;
}
```

ESERCIZIO 2

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S → AY → 1Y → 1YAX → 1AAX → 12AX → 123XB → 123BB → 123pB →123pq

ESERCIZIO 3

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

```
afgch
```

La funzione `main()` inizializza l'array di char `charsA` e la stringa ben formata `charsB` e poi invoca la funzione `merge()`.

La funzione `merge()` alloca dinamicamente spazio di memoria sufficiente a contenere tutti i caratteri contenuti in `charsA` e `charsB`; la dimensione di `charsA` è fornita direttamente dal chiamante, mentre la dimensione di `charsB` è calcolata dinamicamente all'interno del ciclo `while`.
Il ciclo `for` itera lungo tutti i caratteri di `charsA` e `charsB` inserendo all'interno dell'area di memoria allocata dinamicamente prima i caratteri di `charsB`, poi i caratteri di `charsA`. Infatti quando il ciclo `for` giunge all'ultimo carattere della stringa ben formata `charsB`, inizia ad inserire nell'area di memoria allocata dinamicamente i caratteri di `charsA`. Infine la funzione merge inserisce un carattere di terminatore di stringa nell'ultima locazione dell'area di memoria allocata dinamicamente.

La funzione `main()` scrive sullo standard output la stringa ben formata restituita dalla funzione `merge()`, costituita dalla concatenazione dei caratteri presenti in `charsB` e `charsA`.