Fondamenti di Informatica T-1 (A.A. 2011/2012) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 12 Luglio 2012 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Date due liste di interi `l1` e `l2` (non necessariamente della stessa lunghezza), si realizzi una funzione ITERATIVA

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

che restituisca una nuova lista della stessa lunghezza di `l1` contenente valori 1 o 0 a seconda che gli elementi della lista `l1` siano presenti o meno nella lista `l2`. Ad esempio se `l1 = [2, 3, 7, 2, 1, 4, 2]` e `l2 = [2, 1, 10 2]`, la funzione `contains()` deve restituire la lista `[1, 0, 0, 1, 1, 0, 1]` in quanto solo gli elementi `1` e `2` di `l1` sono presenti in `l2`.

La funzione `contains()` 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 `contains()` creata.

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

ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione:

```
char ric(char ch, int i){
      ch = ch + i;
      if( ch > 'f' ){
            return ch;
      }
      else{
            return -1 + ric(ch, i+1);
      }
}
```

Mostrare la sequenza dei record di attivazione e il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali ('c'-2, 1.8F).

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>

#define DIM 10

void change(char in, char out, char* word){

	int size;

	for (size=0; word[size] != '\0'; size++);

	while( size > 0 ){
		if( word[size-1] == in ){
			word[size-1] = out;
		}
		size--;
	}

	return;
}

int main() {
	char *values, *temp;
	char *parola = "carte";
	temp = values = (char*) malloc( sizeof(char) * DIM );
	do{
		*temp = *(parola + 1);
		parola++;
		temp++;
	}
	while( *parola != '\0' );

	printf("prima: %s\n", values);

	change('a', 'e', values);

	printf("dopo: %s\n", values);

	return 0;
}
```

ESERCIZIO 4 (1 punti)

Si illustri come avviene il passaggio dei parametri nell'invocazione di una funzione/procedura in linguaggio C. In particolare, si discuta la differenza tra passaggio dei parametri per valore e per riferimento.

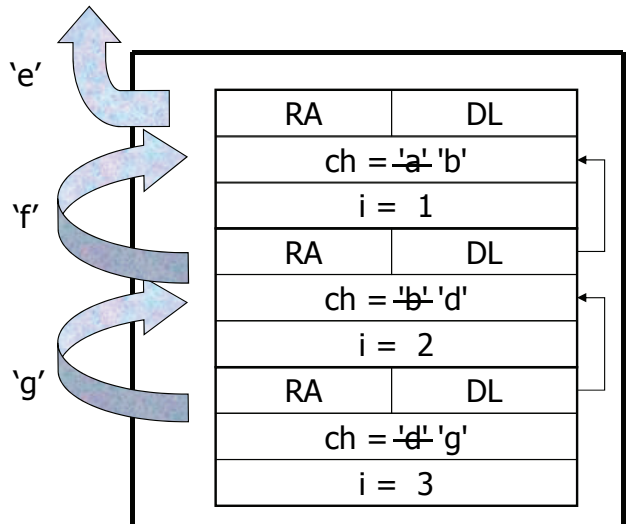
Soluzioni

ESERCIZIO 1

```
list contains(list l1, list l2){
      int found;
      list temp;
      list res = emptylist();
      while( ! empty(l1) ){
            temp = l2;
            found = 0;
            while( !found && !empty(temp) ){
                  if( head(l1) == head(temp) ){
                        found = 1;
                  }
                  temp = tail(temp);
            }
            res = cons(found, res);
            l1 = tail(l1);
      }
      return res;
}

int main(){
      list l1, l2, res;
      l1 = cons(2, cons(3, cons(7, cons(2, cons(1,
                                          cons(4, cons(2, emptylist()))))))));
      l2 = cons(2, cons(1, cons(10, emptylist())));

      printf("contains\n");
      res = contains(l1, l2);
      while( ! empty(res) ){
            printf("%d ", head(res));
            res = tail(res);
      }
      printf("\n");

      return 0;
}
```

ESERCIZIO 2

La funzione restituisce il valore ‘e’.



ESERCIZIO 3

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

```
prima: arte
dopo: erte
```

La funzione `main()` alloca dinamicamente spazio di memoria sufficiente a contenere 10 caratteri e poi assegna a tale area di memoria i caratteri della stringa ben formata `parola` incuso il terminatore di stringa ma escluso il primo carattere. In seguito la funzione `main()` invoca la funzione `change()`.

La funzione `change()` conta il numero di caratteri della stringa ben formata `word` e poi itera su ciascun elemento di tale stringa partendo dal carattere precedente il terminatore di stringa, sostituendo ogni occorrenza del carattere `in` col carattere `out`.

La funzione `main()` stampa sullo standard output la stringa ben formata `values`, sia prima che dopo l’invocazione della funzione `change()`.