Fondamenti di Informatica T-1 (A.A. 2010/2011) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Mercoledì 19 Gennaio 2011 – durata 1h
Totale 12 punti, sufficienza con 7
Compito A

ESERCIZIO 1 (6 punti)

Data una lista di stringhe ben formate `stringhe`, un intero `soglia` e un char `find`, si realizzi una funzione iterativa

```
list occorrenze(list stringhe, int soglia, char find);
```

che restituisca una nuova lista con gli elementi di `stringhe` selezionati in relazione al valore `soglia` e al carattere `find`; in particolare la lista restituita deve contenere i valori della lista `stringhe` solo se contengono il carattere `find` almeno `soglia` volte.

Ad esempio se `stringhe = ["ciao", "a", "tutti", "quanti"]`, `soglia` è 2 e `find = 't'`, la funzione `occorrenze` deve restituire la lista `["tutti"]` poiché `"tutti"` contiene tre `'t'`.

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

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

ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione:

```
float ric(int a, int b){
      if(a<b){
            return 0;
      }
      else{
            return ric(a/b, a*b) + a/b + a*b;
      }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali (2.8, 1.2).

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 DIM 6

char* jump(char* chars, int len, char* jumps){
	int i, tot=0;

	char *res, *temp;
	temp = res = (char*)malloc(sizeof(char)*len);

	while( *jumps != '\0' ){
		tot ++;
		jumps++;
	}
	printf("%d\n", tot);

	for(i=0; i<len; i++){
		if( ! (chars[i]>='0' && chars[i]<='9')){
			*temp = chars[i] + tot;
		}
		else{
			*temp = chars[i] - tot;
		}
		temp++;
	}
	return res;
}

int main(void){
	char j[] = "cba";
	char* res;
	int i;
	char s[DIM] = {'a','5','m','b','3','a'};
	res = jump(s, DIM, j);
	for(i=0; i<DIM; i++){
		printf("%c",*res);
		res++;
	}
	printf("\n");
	return 0;
}
```

ESERCIZIO 4 (1 punti)

Il candidato illustri brevemente la memoria stack ed in quale contesto essa venga utilizzata nella gestione a runtime del linguaggio C.

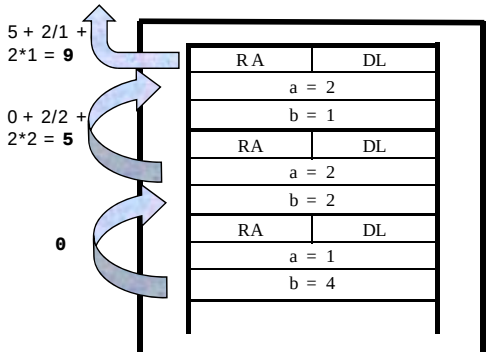
Soluzioni

ESERCIZIO 1

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

list occorrenze(list stringhe, int soglia, char find){
      char *temp; int occ;
      list res;
      res = emptylist();
      while( ! empty(stringhe) ){
            occ = 0;
            temp = head(stringhe);
            while(*temp != '\0'){
                  if( *temp == find ){
                        occ++;
                  }
                  temp++;
            }
            if( occ >= soglia ){
                  res = cons(head(stringhe),res);
            }
            stringhe = tail(stringhe);
      }
      return res;
}

int main(void) {
      list lista, res;
      char c = 't';

      lista = cons("ciao",cons("a",cons("tutti",cons("quanti",emptylist()))));

      res = occorrenze(lista, 2, c);
      while( ! empty(res) ){
            printf("%s\n", head(res));
            res = tail(res);
      }
      return 0;
}
```

ESERCIZIO 2

La funzione restituisce il valore 9.



ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
3
d2pe0d

La funzione `main()` inizializza la stringa ben formata `j` e l'array di char `s` e poi invoca la funzione `jump()`.

La funzione `jump()` alloca dinamicamente spazio sufficiente a contenere `DIM` char.
Il ciclo `while` ha il compito di contare la lunghezza della stringa `jumps` (escluso il terminatore); alla fine del ciclo `while` viene stampato sullo standard output il valore della variabile `tot`, ovvero 3.
Il ciclo `for` itera lungo gli elementi dell'array `chars` ed inizializza l'area di memoria precedentemente allocata. Se l'elemento dell'array `chars` preso in esame non è una cifra, inserisce nell'area di memoria il carattere che si trova in 3 posizioni successive all'elemento in esame, altrimenti il carattere che si trova in 3 posizioni precedenti: 'd' al posto di 'a', '2' al posto di '5', 'p' al posto di 'm', 'e' al posto di 'b', '0' al posto di '3', 'd' al posto di 'a'.
Infine la funzione `jump()` restituisce un riferimento all'area di memoria allocata dinamicamente.

La funzione `main()` scrive sullo standard output il valore degli elementi inseriti nell'area di memoria allocata dinamicamente.