Fondamenti di Informatica T-1 (A.A. 2010/2011) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Mercoledì 14 Settembre 2011 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Data una lista di stringhe ben formate `str` (lista non vuota), si realizzi una funzione ITERATIVA

```
list less(list str);
```

che restituisca una nuova lista contenente gli elementi di `str` che contengono un numero di caratteri numerici inferiore rispetto all'elemento successivo della lista stessa.

Ad esempio se `str = ["mont2ag5na", "m6ar2e4", "c34i8t3ta", "nev8e"]`, la funzione `less()` deve restituire la lista `["mont2ag5na", "m6ar2e4"]`, ovvero i soli valori della lista `str` che rispettano il requisito dato. Infatti `"mont2ag5na"` contiene meno caratteri numerici di `"m6ar2e4"` e `"m6ar2e4"` contiene meno caratteri numerici di `"c34i8t3ta"`, mentre `"c34i8t3ta"` contiene più caratteri numerici di `"nev8e"` e `"nev8e"` non ha un elemento successivo.

A tal scopo si consiglia di realizzare un funzione di supporto `countDigit(...)` che, data una stringa ben formata in ingresso, restituisce il numero di caratteri numerici contenuti in tale stringa ben formata.

La funzione `less()` 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 `less()` creata.

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

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):

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* toChar(float* fl, int* len){
	char *res, ch;
	float *temp;
	int i = 0;
	*len = 0;
	temp = fl;

	while( *temp > 0 ){
		temp++;
		(*len)++;
	}

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

	while( *fl > 0 ){
		ch = 'a' + *fl;
		if( ch >= 'a' && ch <= 'z' ){
			res[i] = ch;
			i++;
		}
		fl++;
	}
	*len = i;

	return res;
}

int main() {
	float array[] = {3.2, 62.0, 3.7, 0.9, -1.5};
	int res, i;
	char* outC;
	outC = toChar(array, &res);
	printf("%d\n", res);
	for(i=0; i<res; i++){
		printf("%c", outC[i]);
	}
	printf("\n");
	return 0;
}
```

ESERCIZIO 4 (1 punti)

Il candidato illustri brevemente le differenze tra allocazione dinamica e statica, realizzando un breve esempio di codice che faccia corretto uso di entrambe le tecniche di allocazione di memoria.

Soluzioni

ESERCIZIO 1

```
int countDigit(char* str){
      int res = 0;
      while( *str != '\0'){
            if( *str>='0' && *str<='9'){
                  res++;
            }
            str++;
      }
      return res;
}

list less(list l1){
      list res = emptylist();
      while( !empty(l1) && !empty(tail(l1)) ){
            if( countDigit(head(l1)) < countDigit(head(tail(l1))) ){
                  res = cons(head(l1), res);
            }
            l1 = tail(l1);
      }
      return res;
}

int main(){
      list l, res;
      l = cons("mont2ag5na", cons("m6ar2e4", cons("c34i9t3ta",
                                                cons("nev8e", emptylist()))));
      res = less(l);
      while( ! empty(res) ){
            printf("%s\n", head(res));
            res = tail(res);
      }
      return 0;
}
```

ESERCIZIO 2

+29 - 77 = -48

+29 --> 00011101
+77 --> 01001101
10110010
-77 --> 10110011

+29 00011101
-77 10110011
-48 11010000 = -128 + 64 + 16 = -48

ESERCIZIO 3

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

```
3
dda
```

La funzione `main()` inizializza l'array di float `array` e poi invoca la funzione `toChar()`.

La funzione `toChar()` conta quanti float sono presenti nell'array `fl` fino al primo valore negativo; in seguito alloca dinamicamente spazio di memoria sufficiente a contenere un numero di caratteri corrispondente ai valori contati (nel caso specifico 4).
Il secondo ciclo `while` itera lungo gli elementi dell'array `fl` fino al primo elemento con valore negativo. Ad ogni iterazione assegna alla variabile `ch` il valore del carattere '`a`' sommato al valore dell'elemento di `fl` (troncato ad intero) attualmente in considerazione. Se il valore risultante della variabile `ch` corrisponde ad un carattere alfabetico minuscolo, allora la funzione `toChar()` memorizza il valore di `ch` nello spazio di memoria allocato dinamicamente. Dopo il secondo ciclo `while`, la funzione `toChar()` assegna alla cella di memoria referenziata dalla variabile `len` il valore di `i`, ovvero il numero di caratteri inseriti nell'area di memoria allocata dinamicamente (nel caso specifico 3). Infine la funzione `toChar()` restituisce al chiamante un riferimento alla prima cella dell'area di memoria allocata dinamicamente.

La funzione `main()` scrive sullo standard output il numero di caratteri inseriti nello spazio di memoria allocato dinamicamente ed i caratteri inseriti in tale area.