Fondamenti di Informatica T-1 (A.A. 2011/2012) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 13 Settembre 2012 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Date due liste di char `l1` e `l2`, si realizzi una funzione RICORSIVA

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

che restituisca una nuova lista contenente i soli elementi presenti in entrambe le liste `l1` e `l2`. Ad esempio se `l1 = ['c', 'a', 'r', 'l', 'o']` e `l2 = ['c', 'a', 'i', 'o']`, la funzione `intersection()` deve restituire la lista `['c', 'a', 'o']` in quanto solo i caratteri `'c'`, `'a'` e `'o'` sono presenti sia in `l1` che in `l2`.

La funzione `intersection()` 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 `intersection()` creata.

Nota: l'ordine degli elementi della lista restituita dalla funzione `intersection()` 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):

21 – 88

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>

char* intreccia(char* str1, char* str2, int *dim){

	char * res;
	int temp;

	for (*dim=0; str1[*dim] != '\0'; (*dim)++);

	for (temp = *dim; str2[*dim-temp] != '\0'; (*dim)++);

	(*dim)++;

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

	*dim = 0;
	while( ! ( *str1 == '\0' && *str2 == '\0' ) ){
		if( *dim % 2 == 0 ){
			res[*dim] = *str1;
			str1++;
		}
		else{
			*(res + *dim) = *str2;
			str2++;
		}
		(*dim)++;
	}
	return res;
}

int main() {
	char *out;
	int x, t=0;
	out = intreccia("mare", "sole", &x);
	printf("%d\n",x);
	for(;t<x;t++){
		printf("%c", out[t]);
	}
	return 0;
}
```

ESERCIZIO 4 (1 punti)

Relativamente al linguaggio C, il candidato illustri le differenze nel tempo di vita delle variabili globali e locali rispetto a una funzione, mostrando anche brevi esempi di codice.

Soluzioni

ESERCIZIO 1

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

int main(){
      list l1, l2, res;
      l1 = cons('c', cons('a', cons('r', cons('l', cons('o', emptylist())))));
      l2 = cons('c', cons('a', cons('i', cons('o', emptylist()))));

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

      return 0;
}
```

ESERCIZIO 2

+ 21 – 88 = -67

+ 21	==>	00010101
+88	==>	01011000
		10100111
-88	==>	10101000
-67	==>	10111101 = -128 + 32 + 16 + 8 + 4 + 1

ESERCIZIO 3

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

```
8
msaorlee
```

La funzione `intreccia()` calcola la dimensione delle stringhe ben formate `str1` e `str2` e poi alloca spazio sufficiente a contenere tutti i caratteri di `str1` e `str2` più uno. In seguito la funzione `intreccia()` inserisce nello spazio di memoria allocato dinamicamente i caratteri di `str1` e `str2`, uno alla volta, secondo il seguente criterio: un carattere della stringa ben formata `str1` nelle posizioni di indice pari, un carattere della stringa ben formata `str2` nelle posizioni di indice dispari. La funzione `intreccia()` restituisce al chiamante per valore un riferimento all'area di memoria allocata dinamicamente e per riferimento il numero di caratteri inseriti in tale area di memoria, ovvero 8.

La funzione `main()` stampa sullo standard output i caratteri presenti nell'area di memoria restituita dalla funzione `intreccia()`; notare che in tale area non è presente una stringa ben formata, ma una sequenza di `char` (non è presente il terminatore di stringa).