

Fondamenti di Informatica T-1 (A.A. 2013/2014) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Venerdì 10 Gennaio 2014 – durata 1h
Totale 12 punti, sufficienza con 7

Compito A

ESERCIZIO 1 (6 punti)

Date due liste di char lChar1 e lChar2 con lo stesso numero di elementi e contenenti solo caratteri alfabetici minuscoli e un intero maxDiff, si realizzi una funzione RICORSIVA

```
list differ(list lChar1, list lChar2, int maxDiff);
```

che inserisca nella lista restituita gli elementi della prima lista solo se distano meno di maxDiff dal rispettivo elemento della seconda lista. Ad esempio se lChar1 = ['h', 'e', 'l', 'l', 'o'], lChar2 = ['m', 'o', 'n', 'd', 'o'] e maxDiff = 5, la funzione differ() deve restituire la lista ['l', 'o']. Ad esempio, il primo carattere 'l' è incluso nella lista restituita in quanto 'l' e 'n' distano tra loro meno di 5 posizioni mentre il secondo carattere 'l' non è incluso nella lista restituita in quanto 'l' e 'd' distano tra loro 5 o più posizioni.

La funzione differ() 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 differ() creata.

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

+ 57 - 115

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>

int* fun(char* str, float* v){
    int * res, i, length;
    char* str2 = str;

    while( *str != '\0' ){
        str++;
    }

    length = str - str2;
    res = (int*) malloc(sizeof(int)*(length+1));

    for(i=length-1; i>=0; i--){
        res[i] = str2[i] + (int)v[i];
    }
    res[length] = '\0';

    return res;
}

int main(){
    int *str;
    char v1[] = {'c', 'd', 'e', '\0'};
    float push[] = {1.2F, 2.7F, 0.9F};

    printf("%s\n",v1);
    str = fun(v1, push);

    while( *str != '\0' ){
        printf("%c",*str);
        str++;
    }

    return 0;
}
```

ESERCIZIO 4 (1 punto)

Si descrivano brevemente le differenze principali tra accesso a file di testo e file binari, fornendo anche semplici esempi di codice in linguaggio C.

Soluzioni

ESERCIZIO 1

```
list differ(list l1, list l2, int maxDiff){
    int diff;
    if( empty(l1) ){
        return l1;
    }
    else {
        diff = head(l1) - head(l2);
        if( diff < 0 ){
            diff = -1 * diff;
        }
        if( diff >= maxDiff ){
            return differ(tail(l1), tail(l2), maxDiff);
        }
        else {
            return cons( head(l1), differ(tail(l1), tail(l2), maxDiff) );
        }
    }
}

int main(){
    list l1, l2, res;

    l1 = cons('h', cons('e', cons('l', cons('l', cons('o', emptylist()))));
    l2 = cons('m', cons('o', cons('n', cons('d', cons('o', emptylist()))));

    printf("differ\n");
    res = differ(l1,l2,5);
    while( ! empty(res) ){
        printf("%c", head(res));
        res = tail(res);
    }

    return 0;
}
```

ESERCIZIO 2

$$+ 57 - 115 = - 58$$

$$\begin{array}{rcl} + 115 & \rightarrow & 01110011 = 64 + 32 + 16 + 2 + 1 \\ & & 10001100 \\ - 115 & \rightarrow & 10001101 \\ + 57 & \rightarrow & 00111001 = 32 + 16 + 8 + 1 \\ - 58 & \rightarrow & 11000110 = -128 + 64 + 4 + 2 \end{array}$$

ESERCIZIO 3

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

```
cde  
dfe
```

La funzione `main()` stampa sullo standard output la stringa ben formata `v1` e poi invoca la funzione `fun()` passando come parametri di ingresso la stringa ben formata `v1` e l'array di float `push`.

La funzione iterativa `fun()` incrementa il valore del puntatore `str` fino al terminatore di stringa e poi alloca dinamicamente spazio di memoria sufficiente a contenere un numero di interi equivalente ai caratteri della stringa ben formata `str2` (incluso il terminatore di stringa). Il ciclo `for` inserisce nell'area di memoria allocata dinamicamente i caratteri della stringa ben formata `str2`, partendo dall'ultimo carattere e sommando a ciascun valore il corrispondente elemento dell'array di float `v` (notare che ciascun float viene troncato ad intero). Infine la funzione `fun()` inserisce il terminatore di stringa (ovvero il valore 0) in fondo all'area di memoria allocata dinamicamente, di cui restituisce un riferimento.

La funzione `main()` stampa sullo standard output i valori presenti nell'area di memoria allocata dinamicamente come `char`, fino al terminatore escluso.