

**Fondamenti di Informatica T-1 (A.A. 2008/2009) - Ingegneria Informatica**  
**Prof.ssa Mello**  
**Prova d'Esame di Venerdì 10 Luglio 2009 – durata 2,30h**

**ESERCIZIO 1 (12 punti)**

Una centralina metereologica memorizza i dati delle rilevazioni di temperatura in un file di testo, in cui ogni riga rappresenta una rilevazione secondo il formato **GG/MM/AAAA T\_min T\_max**, in cui la prima parte rappresenta la data della rilevazione, e gli ultimi due campi sono dei float che memorizzano la temperatura minima e massima registrate in quella giornata. All'inizio del file è presente il numero di rilevazioni effettuate. Il contenuto di un ipotetico file delle rilevazioni potrebbe essere:

3

06/01/2009 0 12

25/04/2009 15 23

24/06/2009 18 38

Dopo aver definito una struttura dati **Rilevazione** atta a contenere le informazioni relative ad una rivelazione giornaliera (data e coppia costituita da temperatura minima e massima), il candidato realizzi una funzione

**Rilevazione\* leggi(FILE\* fp, int\* dim)**

che prenda in input un puntatore a file aperto in lettura e in modalità testo provveda a leggere le informazioni contenute nel file memorizzandole in un vettore allocato dinamicamente della dimensione strettamente necessaria. Al termine della funzione, **dim** deve puntare alla dimensione del vettore di rilevazioni.

Il candidato provveda poi a realizzare un **main(...)** che legga le informazioni presenti nel file **rilevazioni.txt** e provveda a stampare sul file binario **report.dat** il mese in cui è stata rilevata l'escursione termica più alta, nonché il valore di tale escursione (nel caso riportato sopra, il mese sarebbe **6** e l'escursione massima **20**). Si supponga che il file contenga almeno una rilevazione, e che non ci siano due mesi distinti con la massima escursione termica.

**ESERCIZIO 2 (10 punti)**

Una ditta che produce software ha realizzato la seguente funzione:

```
list filter(list content, list mask) {
    if(lunghezza(mask) > lunghezza(content) || !check(mask))
        return emptylist();
    else
        return executeFilter(content, mask);
}
```

Lo scopo è selezionare e restituire in output solo alcuni degli elementi contenuti nella lista di stringhe **content**, in base a criteri specificati nella lista di stringhe **mask**. La funzione controlla nella prima istruzione che la lista **mask** soddisfi due requisiti. Se uno dei due requisiti non è soddisfatto, la funzione restituisce una lista vuota; altrimenti, viene invocata la funzione **executeFilter(...)** che provvede al filtraggio vero e proprio.

In particolare, la lista **mask** non deve eccedere in lunghezza la lista **content** (primo requisito); inoltre la lista **mask** deve contenere solo stringhe **"mantieni"** ed **"elimina"**, e tale condizione viene verificata tramite la funzione **check(...)** (secondo requisito).

Il filtraggio avviene nel modo seguente: se in una certa posizione di **mask** è presente la stringa

- **"mantieni"**, allora l'elemento nella corrispondente posizione in **content** viene mantenuto in uscita;
- **"elimina"**, allora l'elemento nella posizione corrispondente in **content** viene ignorato.

Nel caso in cui **"mask"** sia più corta di **"content"**, tutte le stringhe in eccesso appartenenti alla lista **"content"** vengono automaticamente ignorate.

Ad esempio, data la lista "contenuto" [**"pippo"**, **"paperino"**, **"pluto"**, **"minni"**] e la lista "maschera" [**"mantieni"**, **"elimina"**, **"mantieni"**], il risultato prodotto sarà [**"pippo"**, **"pluto"**].

Il candidato realizzi:

1. La funzione ricorsiva

**int check(list mask)**

che controlli il contenuto di **mask** secondo il criterio sopra specificato (cioè se contiene solo stringhe “**mantieni**” ed “**elimina**”), restituendo VERO se la lista è ben formata, FALSO altrimenti. Per questa funzione non viene data a disposizione nessuna libreria predefinita sulle liste.

2. La funzione ricorsiva

**list executeFilter(list content, list mask)**

che esegua il filtraggio secondo la politica sopra descritta. Per questa funzione si utilizzino le funzioni di libreria predefinite sulle liste di stringhe.

Si rammenta l'uso della funzione **strcmp(...)** per comparare due stringhe.

### **ESERCIZIO 3 (3 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 verifica):

**28 + (-68)**

### **ESERCIZIO 4 (3 punti)**

Si descriva cosa si intende con il termine “programmazione strutturata” e quali sono i suoi principali costrutti.

### **ESERCIZIO 5 (4 punti)**

Si consideri il seguente codice:

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

int mul(int v[], int m1, int m2) {
    int i,s;
    s=0;
    while(*(v+s))
        s++;
    for(i=s-1; i>-1; i--) {
        int m = *(v+i) > 0 ? m1 : m2;
        int c = *(v+i) < 0 ? -* (v+i) : *(v+i);
        *(v+i)=0;
        while(c) {
            *(v+i) += m;
            c--;
        }
    }
    return s; }

int main() {
    int v[] = {-1,7,3,-4,0,6,-2};
    int d = mul(v,3,4);
    int i=0;
    while(i<d)
        printf("%d\n",v[i++]);
}
```

Qualora il compilatore compili questo programma, quali sono i valori stampati a video a tempo di esecuzione? (si motivi la risposta in maniera opportuna)

## Soluzioni

### ESERCIZIO 1

```
typedef struct
{
    int giorno;
    int mese;
    int anno;
} Data;

typedef struct
{
    Data d;
    float tMin;
    float tMax;
} Rilevazione;

Rilevazione* leggi(FILE* fp, int* dim)
{
    Rilevazione* v;
    int i;
    fscanf(fp, "%d", dim);
    v = (Rilevazione*) malloc(*dim*sizeof(Rilevazione));
    for(i=0;i<*dim;i++)
    {
        fscanf(fp,"%d/%d/%d %f %f",
                &v[i].data.giorno,
                &v[i].data.mese,
                &v[i].data.anno,
                &v[i].tMin,
                &v[i].tMax);
    }
    return v;
}

void main()
{
    FILE* fpIn, fpOut;
    int dim, i;
    int meseMaxEscursione;
    float maxEscursione;
    Rilevazione* rilevazioni;

    fpIn = fopen("rilevazioni.txt","r");
    if(fpIn == NULL)
    {
        printf("Errore apertura file di input\n");
        exit(-1);
    }
    rilevazioni = leggi(fpIn, &dim);
    fclose(fpIn);
    meseMaxEscursione = rilevazioni[0].data.mese;
```

```

maxEscursione = rilevazioni[0].tMax - rilevazioni[0].tMin;
for(i=1; i<dim; i++)
{
    float escursione = rilevazioni[i].tMax - rilevazioni[i].tMin;
    if(escursione > maxEscursione)
    {
        maxEscursione = escursione;
        meseMaxEscursione = rilevazioni[i].data.mese;
    }
}
fpOut = fopen("report.dat","wb");
if(fpOut == NULL)
{
    printf("Errore apertura file di output\n");
    exit(-1);
}
fwrite(&meseMaxEscursione,sizeof(int),1,fpOut);
fwrite(&maxEscursione,sizeof(float),1,fpOut);
close(fpOut);
free(rilevazioni);
}

```

## ESERCIZIO 2

```

int check(list mask)
{
    if(mask == NULL)
        return 1;
    else
    {
        if(strcmp(mask->value,"mantieni")==0 || strcmp(mask->value,"elimina")==0)
            return check(mask->next);
        else
            return 0;
    }
}

list executeFilter(list content, list mask)
{
    if(empty(mask))
        return emptylist();
    else
    {
        list tailFilter = executeFilter(tail(content), tail(mask));
        if(strcmp(head(mask),"mantieni") == 0)
            return cons(head(content), tailFilter);
        else
            return tailFilter;
    }
}

```

### ESERCIZIO 3

28 ->	00011100	00011100 ( 28) +
68 ->	01000100	10111100 (-68) =
	10111011	-----
-68 ->	10111100	11011000 (-40)

### ESERCIZIO 5

Il programma è corretto, compila correttamente e stampa:

4

21

9

16

La funzione `mul(int v[], int m1, int m2)` provvede a modificare la porzione del vettore `v` che va dall'inizio fino alla prima occorrenza del valore 0 (escluso), sostituendo ad ogni valore positivo il prodotto tra il valore stesso ed `m1`, e ad ogni valore negativo il valore assoluto del prodotto tra il valore stesso ed `m2`. Inoltre, restituisce la dimensione della porzione modificata.

Il `main(...)` provvede a invocare la funzione `mul(...)` passando come vettore `{-1, 7, 3, -4, 0, 6, -2}` e come fattori `3` (per i valori positivi) e `4` (per i valori negativi). Il vettore viene modificato da `mul(...)` sostituendo `-1` con `4=1*4`, `7` con `21=7*3`, `3` con `9=3*3` e `-4` con `16=4*4`. Il risultato dell'invocazione della funzione è pari a `4`; tale valore viene memorizzato nella variabile `d`, che viene poi utilizzata in un ciclo di stampa dei valori modificati.