

**PROVA SCRITTA DEL MODULO DI  
ELEMENTI DI INFORMATICA  
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA  
26 luglio 2019**

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

**ESERCIZIO 1 (4 punti)**

Rappresentare il valore -211 in complemento a due con dieci bit.

**ESERCIZIO 2 (4 punti)**

Descrivere in modo chiaro e sintetico l'architettura di un moderno sistema operativo.

**ESERCIZIO 3 (25 punti)**

(3 punti) Un'importante azienda automobilistica vuole dotare la centralina di un nuovo modello di un sistema in grado di riconoscere l'identità del guidatore a partire dal suo volto. All'atto dell'accensione del mezzo, una minicamera scatta la foto del viso del guidatore e mediante un algoritmo di riconoscimento facciale genera un punteggio, valore numerico in *floating point* compreso tra 0.0 e 1.0 con precisione alla prima cifra frazionaria. Più alto è il punteggio, maggiore è la somiglianza del volto del guidatore con quello del proprietario dell'automobile.



I progettisti del sistema decidono che il punteggio calcolato debba essere strettamente maggiore di 0.5 per avere una ragionevole certezza dell'identità del guidatore. Questa soglia va però verificata, in modo da accertarsi che non ci siano errori. Perciò, si mette sotto stress il sistema con molti tentativi di accensione dell'auto e i punteggi ottenuti vengono memorizzati, uno per riga, in un file "punteggi.txt".

A questo punto, a voi quali neoassunti si chiede di scrivere un programma in C che, letto il file, stampi a video la più piccola distanza dei punteggi dal valore 0.5 indicato dai progettisti ottenuta durante lo stress test. Se tale distanza risultasse uguale alla distanza più frequente, si dovrà stampare il messaggio "Soglia validata", altrimenti si dovrà stampare "Soglia non validata".

Per il calcolo della distanza  $d$ , i progettisti decidono di utilizzare la formula  $d = |p - 0.5|$ , dove  $p \in [0,1]$  è un dato punteggio prodotto dal sistema di riconoscimento di volti.

Per implementare la vostra soluzione, decidete di memorizzare tutte le distanze da 0.5 dei punteggi letti da file in una lista concatenata di *floating point*. Sviluppate inoltre le seguenti funzioni:

- 1) (4 punti) `leggiPunteggi(nomefile)`: apre un file di nome `nomefile` contenente una sequenza di valori in *floating point* e memorizza le relative distanze da 0.5 in una lista concatenata restituita in uscita.
- 2) (2 punti) `valoreAssoluto(valore)`: restituisce il valore assoluto di un valore numerico in *floating point* fornito in ingresso.
- 3) (8 punti) `estraiFrequente(l)`: restituisce il valore più frequente presente in una lista concatenata `l` di *floating point*.
- 4) (6 punti) `estraiMinimo(l)`: restituisce il valore minimo presente in una lista concatenata `l` di *floating point*.
- 5) (2 punti) `stampaMessaggi(d_min, d_freq)`: stampa a video il valore `d_min` ed i messaggi "Soglia validata" o "Soglia non validata" considerando che `d_min` è la distanza minima e `d_freq` la distanza più frequente dei punteggi dal valore di soglia indicato dai progettisti.

Nota: Le funzioni `CONS`, `HEAD`, `TAIL` e `ISEMPTY` per la gestione della lista di *floating point* sono già implementate. Bonus di 5 punti per chi riuscirà ad implementare ulteriori funzioni per modularizzare il codice.

### Soluzione dell'esercizio 1.

La prima cosa da fare è convertire il valore senza segno 211 con l'algoritmo delle divisioni successive, ottenendo il valore a 8 bit: 11010011. Ad esso vanno aggiunti due 0 nella parte più significativa per avere: 0011010011. Questa stringa corrisponde al decimale 211 rappresentato in complemento a due con dieci bit. Per calcolare il suo opposto, neghiamo i singoli bit della stringa e poi sommiamo 1 per il valore finale, dato dunque da 1100101101. La correttezza del calcolo si verifica sommando quest'ultimo valore al suo corrispondente positivo, per ottenere la stringa 10000000000, con l'1 di overflow come da definizione di complemento a 2.

### Soluzione dell'esercizio 2.

V. dispense del corso.

### Soluzione dell'esercizio 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define N 6 //le distanze possibili da 0.5 sono 6: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5
//posso usare le cifre frazionarie da 0 a 5 per indicizzare in un vettore
//il numero di volte che appaiono
```

```
typedef struct atomoFloat
{
    float d;
    struct atomoFloat* successivo;
} listaFloat;
```

```
//Funzioni per la gestione della lista concatenata di floating point
listaFloat* CONS(listaFloat* l, float v)
{
    listaFloat* n=(listaFloat*)malloc(sizeof(listaFloat));

    n->d=v;
    n->successivo=l;

    return n;
}
```

```
float HEAD(listaFloat* l)
{
    return l->d;
}
```

```
listaFloat* TAIL(listaFloat *l)
{
    return l->successivo;
}
```

```
int ISEMPY(listaFloat* l)
{
    return l==NULL;
}
```

//Funzioni non richieste per modularizzazione

```
void inizializza(int *v)
{
    int i;
    for (i=0; i<N; i++)
        v[i]=0;
}
```

```
int calcolaIndice(float d)
//trasforma un frazionario con precisione a una cifra in un intero tra 0 e 10
//Esempio: se d=0.i → restituisce i
{
    int v;

    d=d*10.;
    v=ceil(d); //uso l'arrotondamento ceil in math.h

    return v;
}
```

```
int trovaMassimo(int *v)
{
    int maxv=v[0],maxi=0;
    int i;

    for (i=1; i<N; i++)
        if (v[i]>maxv)
        {
            maxv=v[i];
            maxi=i;
        }

    return maxi;
}
```

```
float piuFrequente(int* fd)
//trova l'indice corrispondente al massimo di un vettore e lo trasforma
//in un floating point ad una cifra decimale. Esempio: 5==>0.5
{
    int ff;
    float f;

    ff=trovaMassimo(fd);
    f=((float)ff)*0.1;

    return f;
}
```

```

//Funzioni richieste dall'esame
listaFloat* leggiPunteggi(char* nomefile)
{
    FILE *f;
    listaFloat* l=NULL;
    float v;

    f=fopen(nomefile,"r");
    while(!feof(f))
    {
        fscanf(f,"%f",&v);
        v=valoreAssoluto(v-0.5);
        l=CONS(l,v);
    }
    fclose(f);

    return l;
}

float valoreAssoluto(float v)
{
    if (v<0)
        return -v;
    return v;
}

float estraiFrequente(listaFloat *l)
{
    float d;
    int i, fd[N];
    listaFloat *t=l;

    inizializza(fd);

    while(!ISEMPTY(t))
    {
        d=HEAD(t);
        i=calcolaIndice(d);
        //uso la distanza tra due float trasformata in indice di vettore
        //di interi che conterrà le frequenze di ciascuna distanza
        fd[i]++; //fd[i] è il numero di occorrenze della distanza 0.i

        t=TAIL(t);
    }

    return piuFrequente(fd);
}

```

```

float estraiMinimo(listaFloat *l)
{
    float mind=100., d;
    listaFloat *t=l;

    while(!ISEMPTY(t))
    {
        d=HEAD(t);
        if (d<mind)
            mind=d;
        t=TAIL(t);
    }

    return mind;
}

void stampaMessaggi(float d_min, float d_freq)
{
    printf("\nDistanza minima da 0.5: %4.1f\n",d_min);
    if (d_min==d_freq)
        printf("Soglia validata");
    else
        printf("Soglia non validata: la distanza più frequente vale
%4.1f",d_freq);
}

//Funzione principale
int main()
{
    listaFloat* l;
    float dmin, dfreq;

    l=leggiPunteggi("punteggi.txt");
    dfreq=estraiFrequente(l);
    dmin=estraiMinimo(l);
    stampaMessaggi(dmin,dfreq);

    //Qua si possono aggiungere istruzioni per la deallocazione di l

    return 0;
}

```