	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** Appello del 17/07/2023		COGNOME E NOME
	Fila	Colonna	MATRICOLA

- Il presente plico contiene 3 esercizi e 2 domande e **deve essere debitamente compilato con cognome e nome, e numero di matricola**.
- Il tempo a disposizione è di 1 ora e 45 minuti.
- Non separate questi fogli. Scrivete la soluzione solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità. Cancellate le parti di brutta (o ripudiate) con un tratto di penna.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- È possibile scrivere a matita (e non occorre ricalcare al momento della consegna!).
- **Qualsiasi tentativo di comunicare con altri studenti comporta l'espulsione dall'aula**.
- È possibile ritirarsi senza penalità.
- **Non è possibile lasciare l'aula conservando il tema della prova in corso**.
- **Per il superamento dell'esame è necessario dimostrare sufficienti competenze sia in C sia in Matlab, e quindi saper impostare correttamente esercizi in entrambi i linguaggi**.
- **MOLTO IMPORTANTE**: **risposte poco leggibili** (scritte molto piccolo, con calligrafia poco comprensibile, o molto disordinate) **non saranno considerate nella valutazione**.

Esercizio 1 (10 punti)

I risultati di un appello di informatica B possono essere memorizzati in un array di struct di tipo `Esame`. Ogni struct di tipo `Esame` contiene una struct di tipo `Studente` con le informazioni dello studente (`nome`, `cognome`), un array `esercizi` di 5 elementi float, in cui vengono inseriti i voti dei 3 esercizi e delle 2 domande di teoria, un campo intero `voto` con il voto complessivo, il cui valore viene calcolato dal programma che si vuole sviluppare come somma arrotondata dei valori contenuti nel campo `esercizi` (si veda il punto 3), e un campo `info` di tipo stringa, che può assumere i valori “assente”, “ritirato”, “sufficiente”, “insufficiente”, come indicato di seguito.

In linguaggio C:
1) (2.5 punti) Definire le `struct Studente` ed `Esame` e dichiarare una variabile `appello`, che sia un array di elementi di tipo `Esame`, in grado di rappresentare i dati di un appello con **al massimo** 300 studenti iscritti.

2) (2.5 punti) Scrivere il codice per inserire da tastiera i dati relativi a un appello. Il codice deve consentire inizialmente all'utente di inserire il numero di studenti iscritti all’appello, e successivamente di popolare per ogni studente iscritto i campi `nome`, `cognome`, i voti degli `esercizi` e, opzionalmente, la stringa `info` per indicare se uno studente è risultato assente o si è ritirato. Non si effettuino controlli di validità per i dati inseriti.

3) (2.5 punti) Scrivere il codice per calcolare il `voto` di ogni studente iscritto all'appello, non assente e non ritirato. Il voto viene calcolato come somma dei punti degli esercizi con arrotondamento aritmetico (per esempio 17.49 diventa 17, mentre 17.50 diventa 18). Inoltre, il voto è sufficiente se è almeno 18 (dopo l'arrotondamento) e se il voto dei primi due esercizi è almeno 6, o se uno dei due esercizi è almeno 5 ma l’altro è almeno 8. Si scriva nel campo stringa `info` "sufficiente" o "insufficiente" a seconda della condizione.

4) (2.5 punti) Si calcolino e si stampino le seguenti statistiche sull'appello: il totale degli studenti presenti all'appello, il numero di ritirati, e la percentuale dei compiti insufficienti calcolata sul numero dei compiti effettivamente consegnati (quindi studenti presenti e non ritirati).

Soluzione

1)
```
#define MAX_N_STUDENTI 300
#define MAX_DIM_STR 32
#define N_ES 5
struct Studente
{
    char nome[MAX_DIM_STR];
    char cognome[MAX_DIM_STR];
};

struct Esame
{
    Studente studente;
    float esercizi[N_ES];
    int voto;
    char info[MAX_DIM_STR];
};

Esame appello[MAX_N_STUDENTI];
int iscritti_appello = 0;
```

2)
```
int i, k;
do {
    printf("Inserire il numero di iscritti all'appello: ");
    scanf("%d", &iscritti_appello);
while(iscritti_appello > MAX_N_STUDENTI || iscritti_appello < 0);

for(k=0; k< iscritti_appello; k++)
{
    printf("Inserire nome studente\n");
    scanf("%s", appello[k].studente.nome);
    printf("Inserire cognome studente\n");
    scanf("%s", appello[k].studente.cognome);
    for(i = 0; i < 5; i++)
    {
        printf("Inserire il voto dell'esercizio %d\n", i+1);
        scanf("%f", &appello[k].esercizi[i]);
    }
    printf("Inserire assente, ritirato o stringa vuota se lo studente è presente\n");
    scanf("%s", appello[k].info);
}
```

3)
```
float somma, e1, e2;
for(i = 0; i < iscritti_appello; i++)
{
    if(strlen(appello[i].info) == 0)
    {
        somma = 0.0;
        for(j = 0; j < 5; j++) somma += appello[i].esercizi[j];
        appello[i].voto = (int)(somma + 0.5);
        e1 = appello[i].esercizi[0];
        e2 = appello[i].esercizi[1];
        if(appello[i].voto >= 18 && ((e1 >= 6 && e2 >= 6) || (e1 >= 5 && e2 >= 8) || (e2 >= 5 &&
                                    e1 >= 8)))
            strcpy(appello[i].info, "sufficiente");
        else
            strcpy(appello[i].info, "insufficiente");
    }
}
```

4)
```
int presenti = 0, ritirati = 0, insufficienti = 0;
for(i = 0; i < iscritti_appello; i++)
{
    if(strcmp(appello[i].info, "assente") presenti++;
    if(!strcmp(appello[i].info, "ritirato") ritirati++;
    if(!strcmp(appello[i].info, "insufficiente") insufficienti++;
}
printf("Presenti = %d\nRitirati = %d\nPercentuale insufficienti = %f\n",
       presenti, ritirati, 100.0 * (float)insufficienti / (float)(presenti - ritirati));
```

Esercizio 2 (10 punti)

Un'immagine in bianco e nero può essere rappresentata in Matlab come una matrice in cui ogni elemento vale 0 se il corrispondente pixel è nero, oppure 1 se bianco. Il pixel di coordinate x=1, y=1 corrisponde a quello in alto a sinistra, mentre dato `[r,c] = size(matrice)`, il pixel di coordinate x=c, y=r è quello in basso a destra.

Nota: tutti i punti di questo esercizio sono risolvibili senza fare uso di cicli. Eventuali soluzioni che facciano uso di cicli verranno penalizzate.

1) (2.5 punti) Scrivere il codice Matlab per creare una matrice `immagine` in bianco e nero da 100x100 pixel in cui ogni pixel è inizializzato a bianco o nero casualmente.

2) (3 punti) L'operazione di "crop" di una immagine consiste nel rimuovere il bordo, ottenendo una immagine più piccola. Scrivere il codice Matlab per creare una seconda matrice `immagine2` ottenuta dalla matrice `immagine` rimuovendo la prima, la seconda e l'ultima riga, e anche la prima e le ultime 10 colonne, ottenendo una immagine da 89x97 pixel (89 pixel lungo l'asse x, e 97 lungo l'asse y).

3) (3 punti) L'operazione di copia-incolla di una parte di una immagine si può ottenere identificando nell'immagine di partenza la sottomatrice che contiene la parte da copiare e assegnandola a una nuova matrice o a una porzione di matrice di dimensioni adeguate. Operando solo sulla matrice `immagine2`, copiare la regione di dimensione 50x30 pixel (50 pixel lungo l'asse x, e 30 lungo l'asse y) il cui punto in alto a sinistra è il pixel x=15, y=16 e incollarlo, sempre in `immagine2`, nella regione della stessa dimensione il cui pixel in alto a sinistra ha coordinate x=5, y=5.

4) (1.5 punti) Si cerchino le righe della matrice `immagine2` con, rispettivamente, il minore e maggiore numero di pixel a 1 e si rappresentino in un grafico i valori dei pixel di queste due righe nel seguente modo: la riga con meno pixel a 1 sarà rappresentata con una linea continua rossa, l'altra riga con una linea tratteggiata nera (in ascissa si mettano i valori degli indici delle colonne, da 1 a 89). Si assuma esista una sola riga con il minore numero di pixel e una sola con il maggiore numero di pixel.

Soluzione

```
1)
immagine = round(rand(100, 100));

2)
immagine2 = immagine(3:end-1, 2:end-10);

3)
xsize = 50; ysize = 30;
xcopy = 15; ycopy = 16;
xpaste = 5; ypaste = 5;
temp = immagine2(ycopy:ycopy+ysize-1, xcopy:xcopy+xsize-1);
immagine2(ypaste:ypaste+ysize-1, xpaste:xpaste+xsize-1) = temp;

4)
pixelcount = sum(immagine2, 2);
mindex = pixelcount == min(pixelcount);
maxdex = pixelcount == max(pixelcount);
plot(immagine2(mindex, :), 'r');
hold on;
plot(immagine2(maxdex, :), '--k');
hold off;
```

Esercizio 3 (6 punti)

Sia dato il seguente programma C e immaginate che tale programma, una volta compilato, venga eseguito in un solo processo.

```
1)  #include <stdio.h>

2)  int main()
3)  {
4)      float vendite[6];
5)      char s[1024];
6)      int i, j, k;
7)      for(i = 0; i < 6; i++)
8)      {
9)          printf("Inserisci vendite di %d mese/i fa: ", i+1);
10)         scanf("%f", &vendite[i]);
11)     }
12)     strcpy(s, "");
13)     for(i = 5; i >= 0; i--)
14)     {
15)         k = (int) vendite[i];
16)         for(j = 1; j <= k; j++)
17)             strcat(s, "*");
18)         strcat(s, "\n");
19)     }
20)     printf("%s", s);
21)     return 0;
22) }
```

1) (2 punti) Si dica quali linee di codice provocano una chiamata al sistema operativo di qualunque tipo.

2) (2 punti) Considerando le sole chiamate al sistema operativo che causano una interazione con una periferica e assumendo che ogni interazione con periferiche richieda un’attesa, dire quanti passaggi dallo stato di esecuzione allo stato di attesa avvengono durante l'esecuzione dell'intero programma.

3) (2 punti) Descrivere gli stati che attraversa il processo nella riga corrispondente alla prima interazione con il sistema operativo fino all'esecuzione della riga successiva.

Soluzione

1) Le righe 9, 10, 20, 21 causano ognuna una chiamata al sistema operativo. Le righe 9, 10, 20 per interagire con periferiche, la riga 21 in quanto causa la terminazione del processo.

2) Le righe 9 e 10 essendo in un ciclo eseguito 6 volte causano 6 * 2 = 12 passaggi dallo stato di esecuzione allo stato di pronto. La riga 20 causa un ulteriore passaggio, portando il totale a 13.

3) In corrispondenza della prima printf il processo passa dallo stato di esecuzione allo stato di attesa, poi quando la periferica completa la visualizzazione a schermo il processo passa nello stato di pronto, e infine quando lo scheduler sceglie nuovamente il processo esso passa nello stato di esecuzione, potendo quindi giungere alla riga successiva.

Domanda 1 (3 punti)

Spiegare cos'è un riporto perduto, e la sua relazione con l'overflow.

Domanda 2 (3 punti)

Spiegare, aiutandosi con uno o più esempi, perché il seguente frammento di codice C non trova il massimo elemento nel vettore.

```
#define SIZE 16;
int vettore[SIZE];
/* inizializzazione di vettore non inclusa per brevità */
int i, massimo;
massimo = vettore[0];
for(i = 1; i < SIZE; i++)
    if(vettore[i] > vettore[i-1]) massimo = vettore[i];
```