	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** Appello del 18/1/2023		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 30 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)

Le elezioni dello Stato Libero di Bananas si sono svolte di recente. I risultati non sono ancora noti. I dati relativi al numero di votanti sono stati inseriti in una matrice "votazioni" definita nel seguente modo:

```
#define N_CIRC 700

#define N_PART 10

int votazioni[N_CIRC][N_PART];
```

in cui in ogni riga "r" rappresenta una circoscrizione e ogni colonna "c" rappresenta un partito, ossia nella cella (r, c) della matrice "votazioni" è riportato il numero dei cittadini della circoscrizione "r" che hanno votato per il partito "c".

Il sistema di votazione attuale di questo stato prevede che ogni elettore possa votare in un'unica circoscrizione, quella corrispondente alla sua residenza, ed esprimere un solo voto, ogni partito abbia un candidato per ogni circoscrizione e il candidato risulti vincente se ha preso più voti di tutti gli altri in quella circoscrizione (per semplicità, non si prenda in considerazione il caso dei pareggi). Per esempio, se i risultati fossero dati dalla seguente matrice:

5	12	11	7
5	1	2	3
4	30	2	20

si avrebbe che il partito con indice 0 (prima colonna) avrebbe un candidato eletto (avendo vinto nella circoscrizione di indice 1 – seconda riga), il partito con indice 1 (seconda colonna) avrebbe due candidati eletti (prima e terza riga) e i partiti con indice 2 e 3 (terza e quarta colonna) non avrebbero candidati eletti.

Supponendo di avere la matrice "votazioni" definita come sopra e riempita in tutti i suoi elementi (700 circoscrizioni e 10 partiti), si scrivano dei frammenti di codice C per rispondere alle seguenti domande:

Domanda 1.1 (punti 2)

Si supponga che le elezioni in questo stato sono ritenute valide se almeno il 50% + 1 dei votanti è andato ai seggi e che lo Stato Libero di Bananas ha 16 milioni di elettori attivi per queste votazioni. Si scriva a schermo se le elezioni sono valide (messaggio "Le elezioni sono valide" oppure "Le elezioni non sono valide").

Domanda 1.2 (punti 4)

Si crei un vettore con il numero di candidati eletti per ogni partito e si stampi a schermo l'indice del partito vincente, ovvero che ha il maggior numero di candidati eletti (nell'esempio fornito sopra con quattro partiti e tre circoscrizioni avrebbe vinto il partito con indice 1). Si assuma che non si siano verificati casi di parità tra partiti.

Domanda 1.3 (punti 4)

Si assuma ora di considerare un metodo elettorale di tipo proporzionale con sbarramento al 4%, ovvero un partito ottiene un numero di seggi proporzionale al numero di votanti, a patto che abbia preso almeno il 4% del totale dei voti. Si crei un vettore con gli indici dei partiti che hanno superato lo sbarramento, che hanno cioè una percentuale complessiva dei voti rispetto al totale dei votanti maggiore o uguale al 4%. Si stampi a schermo l'indice corrispondente al partito con la percentuale maggiore. Infine, si stampi a schermo se il risultato è consistente con il precedente metodo di elezione (ovvero se il partito che con il metodo a circoscrizioni aveva più candidati è risultato vincente anche con il metodo proporzionale, supponendo che non ci siano stati pareggi in alcun caso).

Soluzione

```
#include <stdio.h>

#define N_CIRC 700
#define N_PART 10
#define QUOTA 16000000/2 /* rappresenta la meta` dei votanti */


void main(){

int votazioni[N_CIRC][N_PART];

long int votanti = 0;

// Domanda 1.1
int i, j;
for (i = 0; i < N_CIRC; i++)
    for (j = 0; j < N_PART; j++)
        votanti += votazioni[i][j];
if (votanti > QUOTA)
    printf ("Le elezioni sono valide");
else
    printf ("Le elezioni non sono valide");

// Domanda 1.2
int eletti[N_PART];
int ind_eletto, max_voti, n_cand_vincitore;
int ind_part_vin;

for (j = 0; j < N_PART; j++)
    eletti[j] = 0;

for (i = 0; N_CIRC; i++) {
    ind_eletto = 0;
    max_voti = votazioni[i][0];
    for (j = 1; j < N_PART; j++)
        if (max_voti < votazioni[i][j]) {
            ind_eletto = j;
            max_voti = votazioni[i][j];
        }
    eletti[ind_eletto]++;
}
n_cand_vincitore = eletti[0];
for (j = 1; j < N_PART; j++)
    if (n_cand_vincitore < eletti[j])
        n_cand_vincitore = eletti[j];

for (j = 0; j < N_PART; j++)
    if (n_cand_vincitore == eletti[j]) {
        printf ("Il partito con indice %d ha vinto", j);
        ind_part_vin = j; //Questo indice serve per la Domanda 1.3
    }
```

```
// Domanda 1.3
#define SBARRAMENTO 4


long int n_voti_per_partito[N_PART];
int ind_partiti_eletti[N_PART];
int count = 0;
int ind_vincitore_propor = -1;
float perc_partito, perc_vincitore_propor = -1;

for (j = 0; j < N_PART; j++)
    n_voti_per_partito[j] = 0;

for (i = 0; i < N_CIRC; i++)
    for (j = 0; j < N_PART; j++)
        n_voti_per_partito[j] += votazioni[i][j];

for (j = 0; j < N_PART; j++) {
    perc_partito = (1.0 * n_voti_per_partito[j]) / votanti * 100;
    if (perc_partito >= SBARRAMENTO) {
        ind_partiti_eletti[count] = j;
        count++;
        if (perc_partito > perc_vincitore_propor) {
            ind_vincitore_propor = j;
             perc_vincitore_propor = perc_partito;

        }
   }

}
printf("L'indice del partito vincitore è: %d", ind_vincitore_propor);

if (ind_vincitore_propor == ind_part_vin)
    printf ("Il risultato è consistente");
else
    printf ("Il risultato è non consistente");
```

Esercizio 2 (10 punti)

Un magazzino è rappresentato da un vettore M, in cui l'elemento i-simo rappresenta la disponibilità attuale in numero di pezzi dell'i-simo prodotto. Un ordine di un cliente è rappresentato da un vettore, delle stesse dimensioni del vettore magazzino, che contiene il numero di prodotti richiesti per ogni categoria. Gli ordini di tutti i clienti della settimana sono invece rappresentati da una matrice, in cui l'elemento (i, j) rappresenta la quantità del prodotto j-simo per l'ordine dell'i-simo cliente. Per esempio, si considerino il vettore M e la matrice O che seguono:

M = [4 12 7 15]

O = [1 0 2 3; 0 4 0 0; 5 0 3 1]

M rappresenta un magazzino in cui sono disponibili 4 unità del primo prodotto, 12 del secondo, 7 del terzo e 15 del quarto. O rappresenta la matrice settimanale degli ordini dei clienti, in cui il primo cliente richiede un'unità del primo prodotto, 2 del terzo e 3 del quarto, il secondo cliente 4 unità del secondo prodotto e il terzo cliente 5 unità del primo prodotto, 3 del terzo e 1 del quarto.

Si sviluppi uno script Matlab che:

Esercizio 2.1 (1.5 punti)

Carichi dal file binario di Matlab **`magazzino.m`** il solo vettore M (il file potrebbe contenere anche altri dati che non si vogliono caricare) e verifichi che tutti i suoi elementi siano validi (ovvero positivi e interi). Solo nel caso in cui tutti gli elementi del magazzino siano validi, salvi in un file **`magazzino_valido.m`** il vettore M.

Esercizio 2.2 (1.5 punti)

Chieda all'utente di inserire una matrice O (con una sola istruzione) e ne controlli la validità (ordini positivi e interi e con dimensioni coerenti con quelle del magazzino). In caso non sia valida si stampi a schermo un messaggio d'errore e si richieda all'utente una matrice di ordini O finché essa non sia valida.

Esercizio 2.3 (4 punti)

Crei un vettore **`richiesta`** di dimensione pari al numero dei prodotti che abbia elementi uguali a zero nel caso in cui il fabbisogno degli ordini possa essere soddisfatto dal magazzino e il numero di unità mancanti nel caso in cui l'ordine non possa essere soddisfatto. Per esempio, nell'esempio fornito il vettore **`richiesta`** sarebbe [2 0 0 0] in quanto nel magazzino M mancherebbero due unità del primo prodotto per soddisfare gli ordini dei clienti in O, mentre la quantità presente degli altri prodotti permette di evadere tutti gli ordini presenti nella stessa matrice O.

Esercizio 2.4 (3 punti)

Disegni un grafico:

- con una linea continua rossa in cui l'ascissa indica i differenti prodotti e l'ordinata il numero di elementi dei prodotti disponibili nel magazzino
- un asterisco verde in corrispondenza della quantità attualmente presente in magazzino solo per quei prodotti che non riescono a soddisfare gli ordini settimanali

Infine, si aggiungano titolo ed etichette agli assi del plot.

Nota: per i punti 2.3 e 2.4 è possibile scrivere una soluzione senza ricorrere all'utilizzo di cicli. Soluzioni che utilizzino cicli verranno penalizzate al momento della correzione.

Soluzione

```
clear
clc
close all


% Esercizio 2.1
load magazzino M
if (M >= 0 & round(M) == M)
    save magazzino_valido M
end


% Esercizio 2.2
O = input("Inserire la matrice degli ordini");
while (any(O < 0 | round(O) ~= O) | size(O, 1) ~= length(M))
    disp("I prodotti inseriti non sono validi");
    O = input("Inserire la matrice degli ordini");
end


%Esercizio 2.3
pezzi_necessari = sum(O);
richiesta = (pezzi_necessari - M) .* (pezzi_necessari > M);


%Esercizio 2.4
figure();
prodotti = 1:length(M);

plot(prodotti, M, 'r');
hold on
plot(prodotti(pezzi_necessari > M), M(pezzi_necessari > M), 'g*');
title("Magazzino");
xlabel("Prodotti");
ylabel("Quantità");
```

Esercizio 3 (6 punti)

In un sistema monoprocessore (con un'unica unità di elaborazione) con sistema operativo multiprogrammato, i processi vengono gestiti secondo una politica di turnazione circolare senza priorità (chiamata round-robin) con un quanto di tempo di 20 microsecondi. Tale sistema viene utilizzato per un'applicazione di risoluzione di equazioni di fluidodinamica che comporta l'esecuzione di 10 processi P1, ..., P10, di durata simile, che possono essere eseguiti in parallelo. Ciascuno di questi processi interagisce con l'ambiente esterno caricando da file i dati di lavoro all'inizio della sua esecuzione e salvandoli su file solo al termine dell'esecuzione. Queste ultime due operazioni richiedono l'interazione del processo con il disco rigido del calcolatore. Si assuma che il tempo di calcolo effettivo necessario per l'esecuzione di ciascuno di questi processi è dell'ordine dei secondi.

Esercizio 3.1 (2.5 punti)

Indicare in quali stati passano durante la loro esecuzione i processi P1, …, P10.

Esercizio 3.2 (2 punti)

Vi viene proposto di passare a un nuovo sistema con un quanto di tempo molto inferiore, ad esempio un quanto di tempo di 1 microsecondo. Motivare se la scelta potrebbe migliorare le performance in termini di tempo di esecuzione medio per l'applicazione precedente (non sono necessari calcoli). In particolare, si discuta anche se altre caratteristiche della gestione dei processi possano influenzare la scelta.

Esercizio 3.3 (1.5 punti)

Supponendo che ognuno dei processi P1, ..., P10 richieda sul calcolatore su cui viene eseguito circa un secondo di calcolo effettivo, qual è il numero approssimativo di passaggi dallo stato di pronto allo stato di esecuzione che si verificherebbero in ogni processo nel caso in cui il quanto di tempo sia pari a 20 microsecondi? Quanti nel caso in cui il quanto di tempo sia pari a 1 microsecondo?

Soluzione

Esercizio 3.1 Poiché i processi P1, …, P10 richiedono interruzioni interne solo all'inizio e alla fine della loro esecuzione (per caricare e salvare i dati, il che necessita di andare nello stato di attesa), passeranno dallo stato di pronto a quello di esecuzione un numero di volte approssimativamente pari al rapporto tra il tempo totale di esecuzione di ciascuno di essi e il quanto di tempo definito dal sistema operativo (20 microsecondi in questo caso). Per esempio, assumendo che il tempo di ogni processo sia molto maggiore di 20 microsecondi, avremo: esecuzione di P1, caricamento dati P1 e passaggio in stato di attesa, esecuzione di P2, caricamento dati P2 e passaggio in stato di attesa, …, esecuzione di P10, caricamento dati P10 e passaggio in stato di attesa, esecuzione di P1 per 20 microsecondi, context switch (P1 torna in pronto), esecuzione di P2 per 20 microsecondi, context switch (P2 torna in pronto), esecuzione di P3 per 20 microsecondi, …

Non appena ciascuno dei processi sarà prossimo al termine dell'esecuzione, dovrà fare un ulteriore passaggio da esecuzione ad attesa per il salvataggio dei dati. Dopo questa operazione il processo stesso passerà negli stati di pronto ed esecuzione prima di terminare.

Esercizio 3.2 Se l'applicazione si intende completata quando tutti i processi sono stati eseguiti, l'adozione del secondo sistema non produrrà degli effetti positivi rispetto al primo.

In particolare, diminuendo il quanto di tempo di esecuzione andremmo ad aumentare il numero dei context switch eseguiti dal sistema, introducendo ulteriore ritardo nella computazione. Quindi, nel caso in cui il tempo richiesto per i context switch siano analoghi nei due sistemi, avremmo che il primo sistema darà delle performance leggermente migliori.

Esercizio 3.3 Poiché 1 s = 1 * 10^6 microsec, avremo:

Caso 1: numero passaggi da pronto a esecuzione: 10^6/20 + 2 (per tenere conto dei passaggi di stato che avvengono dopo le interruzioni interne necessarie per gestire l'input/output)

Caso 2: numero passaggi da pronto a esecuzione: 10^6 + 2

Nota: anche una soluzione approssimativa che non considera l'input/output sarebbe accettabile.

Domanda 1 (3 punti)

Si consideri il numero –347. Si dica quali tra le seguenti affermazioni sono corrette (è possibile che più affermazioni, anche tutte, siano corrette) e perché:

- Il numero è rappresentabile in complemento a due utilizzando 9 bit
- Il numero è rappresentabile in complemento a due utilizzando 12 bit
- 1010100101 è una rappresentazione in complemento a due corretta per il numero –347

Risposte prive di giustificazione non verranno considerate.

Domanda 2 (3 punti)

Finita l'università vi viene chiesto di sviluppare un programma che inseriti i dati relativi alle piante presenti in un edificio, vi avverta ogni qualvolta sia necessario che esse vengano annaffiate. Il committente vi chiede che il programma sia sviluppato al più presto perché deve essere pronto prima che le piante di un grande vivaio muoiano. In quale linguaggio di programmazione scegliereste di sviluppare il programma: C o Matlab? Descrivere le differenze tra i due linguaggi di programmazione e motivare la scelta elencando le criticità della soluzione scelta. N.B. dire esplicitamente se sia necessario aggiungere dettagli per meglio definire il problema e motivare la soluzione corrispondente.