	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** Appello del 8/2/2023		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)

Un parcheggio per automobili rappresenta all'interno del suo sistema informativo i posti disponibili come una matrice bidimensionale di R righe e C colonne, in cui ciascuna riga corrisponde a un settore e ciascuna colonna corrisponde a un posto macchina. Dunque, l'elemento in posizione i, j rappresenta il j-esimo posto macchina nel i-esimo settore. Il valore di ciascun elemento della matrice è 0 (se il parcheggio è libero) oppure 1 (se il parcheggio è già occupato da una macchina). Una serie di sensori aggiornano i valori della matrice ogni volta che una macchina viene parcheggiata o rimossa.

1) **[1 punto]** definire opportunamente una matrice che contenga le informazioni richieste e inizializzarla per rappresentare il parcheggio completamente libero.

2) **[2 punti]** Scrivere un frammento di codice che chieda all'utente di inserire le coordinate di un posto auto (riga e colonna) e, dopo avere controllato la correttezza degli input, stampi a video se il parcheggio è libero od occupato.

3) **[3 punti]** Scrivere un frammento di codice che calcoli il numero totale di posti liberi in un dato momento e lo stampi a video.

4) **[4 punti]** Per agevolare la fruizione del parcheggio si vogliono indirizzare gli automobilisti verso il settore (righe) più vuoto. Scrivere un frammento di codice che individui il settore più libero, ovvero quello con il minore numero di posti auto occupati (in caso di pareggi, restituisca uno qualsiasi dei settori più liberi) e ne stampi a schermo il numero (indice di riga della matrice).

Soluzione

1)
```
#define ROW 5
#define COL 10

int main() {

     // dichiarazione matrice parcheggio
     int parcheggio[ROW][COL];

     // inizializzazione di parcheggio completamente libero
     int i, j;
     for (i = 0; i < ROW; i++)
          for (j = 0; j < COL; j++)
               parcheggio[i][j] = 0;
}
```

2)
```
do{
     printf ("inserire il numero di riga e di colonna: ");
     scanf ("%d %d", &i, &j);
} while (! ((i >= 0 && i < ROW) && (j >= 0 && j < COL)));

if (parcheggio[i][j]) {
     printf ("Posto occupato");
}
else {
     printf ("Posto libero");
}
```

3)
```
int liberi = 0;
for (i = 0; i < ROW; i++)
     for (j = 0; j < COL; j++)
     if (! parcheggio[i][j])
     liberi++;
printf ("Ci sono %d posti attualmente liberi\n", liberi);
```

4)
```
int occupati = 0;
int minimo;
int settore;

for (j = 0; j < COL; j++)
     if (parcheggio[0][j])
          occupati++;
minimo = occupati;
settore = 0;

for (i = 1; i < ROW; i++) {
     occupati = 0;
     for (j = 0; j < COL; j++)
          if (parcheggio[i][j])
```

```
                occupati++;
        if (occupati < minimo) {
             minimo = occupati;
             settore = i;
        }
}
printf("Il settore più libero è il numero %d\n", settore);
```

Esercizio 2 (10 punti)

La piattaforma PolimiFlight permette ai suoi clienti di confrontare l'andamento dei prezzi dei voli aerei, al fine di aiutarli a trovare l'opzione di viaggio più economica in un determinato periodo di tempo. Ad esempio, un utente può inserire la durata della sua permanenza a New York e il budget massimo a sua disposizione per viaggiare nel mese di gennaio, e la piattaforma gli suggerirà la combinazione di date per i voli di andata e ritorno compatibili con il budget indicato.

Ai fini di questo esercizio, si assuma di prendere in considerazione solo un mese dell'anno (gennaio), e che le date di partenza e di arrivo cadano nel medesimo mese.

1. **[4 punti]** Si definisca in linguaggio Matlab la funzione **`trovaOpzioni`** che riceve in ingresso gli array **`prezzi_andata`** e **`prezzi_ritorno`** (che contengono rispettivamente i prezzi dei voli di andata e di ritorno per ogni giorno del mese di gennaio), una **`durata`** della permanenza (in giorni) e un valore **`max_budget`** (massimo budget a disposizione dell'utente). Tale funzione:
    a. dovrà restituire un array **`opzioni_viaggio`**, contenente i prezzi totali di andata e ritorno corrispondenti a tutte le date di partenza possibili che permettano all'utente di ritornare nello stesso mese (a parità di durata del viaggio), e un array **`opzioni_valide`**, ossia una maschera di valori logici che permetta di filtrare le sole opzioni di viaggio compatibili con il budget a disposizione;
    b. dovrà mostrare il grafico dei prezzi totali di tutte le opzioni di viaggio; tale grafico mostrerà sulle ascisse la data di partenza (come numero intero), mentre sulle ordinate il relativo prezzo del viaggio (totale di andata e ritorno) che inizia in tale data;
    c. mostrerà inoltre, sul medesimo grafico, le opzioni di viaggio che soddisfano i requisiti di budget dell'utente, indicandole con dei cerchi rossi.

2. **[6 punti]** Si scriva quindi uno script che, dopo aver opportunamente pulito il workspace:
    a. legga da file i prezzi dei voli per il mese di gennaio; i vettori **`prezzi_andata`** e **`prezzi_ritorno`** conterranno rispettivamente i prezzi dei voli di andata e i prezzi dei voli di ritorno per ogni giorno del mese (31 giorni);
    b. chieda all'utente di inserire il massimo budget a sua disposizione per il viaggio in questione; il budget dovrà coprire il costo totale di andata e ritorno; si chieda all'utente di inserire il budget fino a quando non viene inserito un valore valido (scalare e positivo);
    c. chieda all'utente di inserire la durata del viaggio, fino a quando non inserisce un valore valido (valore scalare tra 1 e 30). Immaginiamo che la data di partenza e quella di ritorno non debbano coincidere, e che entrambe le date cadano in gennaio (esempio: durata = 3 indica che si parte in una data e si torna 3 giorni dopo);
    d. invochi la funzione **`trovaOpzioni`** definita al punto precedente;
    e. indichi all'utente se esistono opzioni di viaggio valide, e in caso affermativo gli comunichi quante sono e l'opzione più economica in assoluto, indicando il prezzo complessivo e le date di partenza e ritorno (per semplicità, si assuma che non esistano due opzioni di viaggio con il medesimo prezzo).

Soluzione

1.

```
function [opzioni_viaggio, opzioni_valide] = trovaOpzioni(prezzi_andata, prezzi_ritorno, durata,
                                                          max_budget)
       % La funzione trovaOpzioni riceve in ingresso gli array prezzi_andata e
       % prezzi_ritorno (che contengono rispettivamente i prezzi dei voli di andata
       % e di ritorno per ogni giorno del mese di gennaio), una durata della
       % permanenza (in giorni) e un valore max_budget (massimo budget a disposizione
       % dell'utente).
       % La funzione ritorna un array opzioni_viaggio, contenente i prezzi totali di andata e
       % ritorno per tutte le date di partenza possibili che permettano all'utente di ritornare
       % nello stesso mese (a parità di durata del viaggio), e un array opzioni_valide, ossia
       % una maschera di valori che permetta di filtrare le sole opzioni di viaggio compatibili
       % con il budget a disposizione

       % a.
       % Costruisco il vettore con tutte le opzioni di viaggio valide: vado quindi a
       % considerare tutte le date di partenza possibili che mi permettano, a parità di
       % durata del viaggio, di ritornare nello stesso mese
       date_partenze_possibili = 1:length(prezzi_andata)-durata;
       date_ritorno_possibili = durata+1:length(prezzi_ritorno);
        opzioni_viaggio = prezzi_andata(date_partenze_possibili) + prezzi_ritorno(date_ritorno_possibili);

       % Cerco le opzioni di viaggio con costo non superiore al budget indicato dall'utente
       opzioni_valide = opzioni_viaggio <= max_budget;

       % b.
       % Mostro il grafico dei prezzi totali delle varie opzioni di viaggio: sulle
       % ascisse avrò la data di partenza, sulle ordinate il relativo prezzo del
       % viaggio
       hold on;
       plot(date_partenze_possibili, opzioni_viaggio);

       % c.
       % Mostro nel medesimo grafico le opzioni di viaggio che soddisfano i requisiti
       % di budget dell'utente, indicandole con dei cerchi rossi
       date_opzioni_valide = find(opzioni_valide);
       prezzi_opzioni_valide = opzioni_viaggio(opzioni_valide);
       plot(date_opzioni_valide, prezzi_opzioni_valide, 'ro');
       hold off;

     end
```

2.

```
clear
clc
close all

% a.
% Leggo da file i prezzi dei voli per il mese di gennaio: i vettori
% conterranno rispettivamente i prezzi dei voli di andata e i prezzi dei voli
% di ritorno per ogni giorno del mese (31 giorni)
load prezzi_voli.mat

% b.
% Chiedo all'utente di inserire il massimo budget a sua disposizione per il
% viaggio in questione. Il budget dovrà coprire il costo di andata e ritorno.
% Si chieda all'utente di inserire il budget fino a quando non viene inserito un
% valore valido
budget_valido = false;
while ~budget_valido
  max_budget = input('Quale è il massimo budget che hai a disposizione per il viaggio?');

  budget_valido = isnumeric(max_budget) & length(max_budget) == 1 & max_budget > 0;

  if ~budget_valido
    disp('inserisci un budget valido!');
```

```
  end
end

% c.
% Chiedo all'utente di inserire la durata del viaggio, fino a quando non
% inserisce un valore valido. Immaginiamo che la data di partenza e quella di
% ritorno non debbano coincidere, e che entrambe le date cadano in gennaio
% Esempio: durata = 3 indica che si parte in una data e si torna 3 giorni dopo
durata_valida = false;
while ~durata_valida
  durata = input('Quale è la durata della permanenza?');

  durata_valida = isnumeric(durata) & length(durata) == 1 & durata > 0 & durata < 31;

  if ~durata_valida
    disp('inserisci una durata valida!');
  end
end

% d.
[opzioni_viaggio, opzioni_valide] = trovaOpzioni(prezzi_andata, prezzi_ritorno, durata, max_budget);

% e.
% Verifico se esistono opzioni valide
if ~any(opzioni_valide)
  disp('Non esiste alcuna opzione di viaggio che rispetti durata e budget.');
else

  % Comunico all'utente il numero di opzioni valide trovate
  totali_opzioni_valide = sum(opzioni_valide);
 disp(['Esistono ' num2str(totali_opzioni_valide) ' opzioni che soddisfano i criteri di ricerca']);

  % Mostro all'utente l'opzione migliore trovata
  % Nota: per semplicità, si assuma che non esistano due opzioni di viaggio con
  % il medesimo prezzo

  % Trovo l'opzione migliore e la mostro a video
  [prezzo, data_partenza] = min(opzioni_viaggio);
  data_ritorno = data_partenza + durata;
 disp(['Migliore opzione trovata: dal ' num2str(data_partenza) ' al ' num2str(data_ritorno) '
       gennaio, prezzo: ' num2str(prezzo)]);
end
```

Esercizio 3 (6 punti)

Si eseguano in complemento a due su 8 bit le seguenti operazioni, riportando i calcoli. Si dica se la somma è corretta, se vi sono eventuali overflow e riporti perduti:

1. $115_{dec} + 64_{dec}$
2. $- 97_{dec} + (- 81_{dec})$

Soluzione

1.

```
          1
115dec  ->  0 1 1 1 0 0 1 1 C2
64dec   ->  0 1 0 0 0 0 0 0 C2
      ----------------------
          1 0 1 1 0 0 1 1
```

C'è un overflow, ma nessun riporto perduto, la somma non è corretta.

2.

```
        1   1 1 1 1 1 1
-97dec  ->  1 0 0 1 1 1 1 1 C2
-81dec  ->  1 0 1 0 1 1 1 1 C2
      ---------------------
          0 1 0 0 1 1 1 0
```

C'è overflow con riporto perduto; la somma non è corretta.

Domanda 1 (3 punti)

Supponete di fare le seguenti operazioni su un pc:

1. scaricare la posta arrivata nella vostra casella mail e salvare un file che vi è stato spedito
2. eseguire uno script Matlab che faccia visualizzare a schermo un risultato
3. comunicare tramite un programma di messaggistica istantanea il risultato ottenuto

Per ognuna di queste operazioni si dica quali elementi del kernel del sistema operativo del PC sono stati coinvolti, in che modo e perché. Anche altri elementi del sistema operativo al di fuori del kernel sono stati utilizzati per queste operazioni?

Domanda 2 (3 punti)

Descrivere quali sono le differenze concettuali ed effettive tra le seguenti due istruzioni in C:

- struct {…} esempio;
- typedef struct {…} Esempio;

Proporre un esempio in cui la prima sintassi risulti più adatta e un esempio in cui sia più adatta la seconda.