	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** Appello del 28/06/2022	COGNOME E NOME MATRICOLA
		Spazio riservato ai docenti

- 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!).
- **È vietato utilizzare calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- **È consentita la consultazione di libri, appunti, slide e qualsiasi altra risorsa**.
- **Qualsiasi tentativo di comunicare con altri studenti, o se trovate risposte alle domande d'esame che riportano soluzioni copiate da altre persone/fonti, 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'agenzia immobiliare di Milano gestisce al più 100 appartamenti sia in affitto che in vendita. Un appartamento è caratterizzato da: indirizzo (una stringa di al più 30 caratteri), numero di stanze, prezzo e tipologia (affitto oppure vendita). Nel caso di appartamenti in affitto il prezzo è da intendersi come canone di locazione mensile, mentre nel caso di appartamenti in vendita il prezzo è da intendersi come costo dell'appartamento. Svolgere, utilizzando il linguaggio C, i seguenti punti:

1. (3 punti) Definire le strutture dati necessarie a gestire l'agenzia immobiliare.

2. (4 punti) Supponendo che le strutture dati per gestire l'agenzia siano state precedentemente riempite correttamente, scrivere un frammento di codice che chieda all'utente di quante stanze ha bisogno (controllando che il valore inserito sia corretto) e cerchi l'appartamento in vendita più economico che abbia esattamente un numero di stanze pari a quello indicato dall'utente; se esistono più appartamenti con il numero di stanze desiderato aventi prezzo minimo, se ne consideri uno a piacimento. Dell'appartamento individuato si stampi a video l'indirizzo e il prezzo. Se nessun appartamento ha il numero di stanze indicato dall'utente, si stampi a video un messaggio che informi l'utente di ciò.

3. (3 punti) Scrivere un frammento di codice che permetta a un operatore dell'agenzia di inserire i dati di un nuovo appartamento (se l'agenzia non ha già raggiunto il numero massimo di appartamenti gestibili). Tutti i dati relativi al nuovo appartamento devono essere inseriti da tastiera dall'operatore e la loro correttezza deve essere controllata.

SOLUZIONE

Punto 1

```
#define LEN_INDIRIZZO 30 + 1
#define N_MAX_APPARTAMENTI 100

typedef enum {affitto, vendita} tipo_appartamento;

typedef struct {
        char indirizzo[LEN_INDIRIZZO];
        int n_stanze;
        int prezzo;
        tipo_appartamento tipo;
} appartamento;

typedef struct {
        appartamento gestiti[N_MAX_APPARTAMENTI];
        int n_appartamenti;
} agenzia_immobilire;

agenzia_immobilire agenzia;
```

Punto 2

```
int quante_stanze;
int primo = 1, i, i_min = -1;

do {
        printf (“Quante stanze si desiderano?\n”);
        scanf (“%d”, &quante_stanze);
} while (quante_stanze <= 0);

for (i = 0; i < agenzia.n_appartamenti; ++i) {
        if (agenzia.gestiti[i].tipo == vendita && agenzia.gestiti[i].n_stanze == quante_stanze) {
                if (primo == 1) {
                        i_min = i;
                        primo = 0;
                }
                else {
                        if (agenzia.gestiti[i].prezzo < agenzia.gestiti[i_min].prezzo)
                                i_min = i;
                }
        }
}

if (i_min == -1) {
        printf (“Non è stato trovato alcun appartamento con %d stanze\n”, quante_stanze);
}
else {
        printf (“L’appartamento in vendita più economico avente %d stanze si trova in %s e costa %d\n”,
quante_stanze, agenzia.gestiti[i_min].indirizzo, agenzia.gestiti[i_min].prezzo);
}
```

Punto 3

```
char indirizzo[LEN_INDIRIZZO], tipo;
int prezzo, n_stanze;
if (agenzia.n_appartamenti < N_MAX_APPARTAMENTI) {
        do {
                printf (“inserisci il prezzo dell’appartamento\n”);
                scanf (“%d”, &prezzo);
```

```
            fflush (stdin);
        } while (prezzo <= 0);

        do {
            printf (“inserisci il tipo dell’appartamento (‘a’ per affitto, ‘v’ per vendita)\n”);
            scanf (“%c”, &tipo);
            fflush (stdin);
        } while (tipo != ‘a’ && tipo != ‘v’);

        do {
            printf (“inserisci il numero di stanze dell’appartamento\n”);
            scanf (“%d”, &n_stanze);
            fflush (stdin);
        } while (n_stanze <= 0);

        printf(“Inserisci l’indirizzo dell’appartamento\n”);
        gets(indirizzo);

        strcpy (agenzia.gestiti[agenzia.quanti].indirizzo, indirizzo);
        agenzia.gestiti[agenzia.quanti].prezzo = prezzo;
        agenzia.gestiti[agenzia.quanti].n_stanze = n_stanze;
        if (tipo == ‘a’)
            agenzia.gestiti[agenzia.quanti].tipo = affitto;
        else
            agenzia.gestiti[agenzia.quanti].tipo = vendita;

        agenzia.quanti++;
}
```

Esercizio 2 (10 punti)

Parte 2.A (5 punti)
Si vuole sviluppare un sistema di controllo del movimento di un robot. Lo spazio che il robot deve esplorare è rappresentato come una matrice logica. Per esempio, la seguente matrice m descrive una zona che è divisa in 10x10 celle. La presenza di un ostacolo in una zona è rappresentata dalla presenza nel valore 0 nella cella corrispondente, mentre la possibilità di transitare per quella zona è rappresentata dalla presenza del valore 1.

	1	2	3	4	5	6	7	8	9	10
1	1	1	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	1	0	1	1	1	0	1	1	0	0
4	1	0	1	0	1	0	0	0	0	0
5	1	1	1	0	1	0	0	0	0	0
6	0	1	1	1	1	0	0	0	0	0
7	0	0	0	0	1	1	0	0	0	1
8	0	0	0	0	0	1	1	1	0	1
9	0	0	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	0	0	0	0

Si sviluppi in linguaggio Matlab la funzione *puoSpostarsiOriz()* che prende come parametri una matrice `m` come quella qui sopra illustrata, che rappresenta la zona da esplorare, e un array contenente i due valori (posizione di riga e di colonna nella matrice) che rappresentano la posizione di partenza del robot. La funzione restituisce al chiamante il numero di spostamenti (espressi come numero di celle della matrice) che il robot può eseguire orizzontalmente verso destra, a partire dalla sua posizione di partenza, tenendo conto che la presenza di ostacoli impedisce lo spostamento. Per esempio, se il robot parte dalla cella (3, 3), la funzione restituirà il valore 2 perché, in orizzontale, il robot può arrivare solo fino alla cella (3, 5) a causa della presenza dell'ostacolo in posizione (3, 6). Se il robot parte dalla cella (3, 8), la funzione restituirà il valore 0 perché non può eseguire nessuno spostamento a destra.
Si faccia in modo, infine, che, se la funzione riceve valori non coerenti (per esempio, una posizione di partenza che si trova fuori dalle dimensioni della matrice), restituisca il valore -1.

Parte 2.B (5 punti)
Si scriva uno script in linguaggio Matlab che svolga le seguenti operazioni:

- Generazione di una matrice di valori zero e uno, con valori disposti in modo casuale.

- Richiesta all'utente di inserire gli indici di riga e colonna che rappresentano la cella di partenza del robot. Si sviluppi questa parte in modo tale che nel caso in cui i valori non siano coerenti con le dimensioni della matrice, lo script torni a ripetere la richiesta per l'utente.

- Chiamata della funzione *puoSpostarsiOriz()* e stampa a video del risultato ottenuto.

Soluzione

Parte 2.A

```
function [nCelle] = puoSpostarsiOriz(matr, start)
% puoSpostarsiOriz controlla se, a partire da una posizione data, un
% robot puo` spostarsi orizzontalmente, da sinistra a destra nello spazio
%
% Data creazione: 24/06/2020. Autori: i docenti di Informatica B
%
% Parametri:
%    matr:   matrice che rappresenta la zona da esplorare
%    start:  array contenente i due valori che rappresentano la posizione di
%            partenza del robot.
%    nCelle: numero di spostamenti che il robot può eseguire orizzontalmente
%            verso destra, a partire dalla sua posizione di partenza, tenendo conto
%            che la presenza di ostacoli impedisce lo spostamento. Assume il valore -1
%            nel caso in cui i valori ricevuti dalla funzione non sono corretti.

[a, b] = size(matr);
 if start(1) > a || start(2) > b || start(1) < 1 || start(2) < 1 || ...
    matr(start(1), start(2)) ~= 1
     nCelle = -1;
 else
     nCelle = 0;
     while(start(2) < b && matr(start(1), start(2)+1) == 1)
         start(2) = start(2)+1;
         nCelle = nCelle+1;
     end
end
```

Parte 2.B

```
% Script che genera la matrice che rappresenta l'area di lavoro per un robot
% e chiama la funzione puoSpostarsiOriz.
%
% Data creazione: 24/06/2022. Autori: I docenti di Informatica B
%
% Variabili importanti:
%    matr: matrice di zero e uno
%    start: posizione di partenza del robot
%    ris: numero di spostamenti che il robot puo` eseguire in orizzontale

dim = input('inserisci la dimensione della griglia: ');
matr = round(rand(dim))

start(1) = input('inserisci il numero della riga da cui parte il robot: ');
while start(1) < 1 || start(1) > dim
  start(1) = input('inserisci il numero della riga da cui parte il robot: ');
end
```

```
start(2) = input('inserisci il numero della colonna da cui parte il robot: ');
while start(2) < 1 || start(2) > dim
  start(2) = input('inserisci il numero della colonna da cui parte il robot: ');
end

ris = puoSpostarsiOriz(matr, start);
fprintf('il robot puo` spostarsi in orizzonale di %d celle\n', ris);
```

Esercizio 3 (6 punti)

Supponendo di lavorare su una macchina a 8 bit:

1. (3 punti) Convertire da decimale a complemento a 2 i seguenti numeri interi:

    A = 56

    B = -43

    C = -87

2. (3 punti) calcolare (lavorando direttamente in binario) D = A + C ed E = B + C mostrando la rappresentazione in complemento a 2 di D e di E d indicando se D ed E possono essere correttamente rappresentati con 8 bit.

SOLUZIONE

A = 00111000

B = 11010101

C = 10101001

D = A + C = -31 = 11100001 OK

E = B + C = -130 = **1**01111110 NO: occorrono 9 bit!

Domanda 1 (3 punti)

È possibile affermare che l'utilizzo del linguaggio Matlab sia più semplice per il programmatore rispetto all'utilizzo del linguaggio C. Si fornisca una spiegazione **sintetica** ma **chiara** delle motivazioni tecniche, relative alle caratteristiche dei due linguaggi, a supporto di questa affermazione.

Domanda 2 (3 punti)

È possibile che in un calcolatore ci siano più processi in stato di attesa e nessuno in stato di esecuzione? Se sì, quando questo si può verificare?