	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** Appello del 06/09/2022		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 2 ore.
- 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!).
- Durante l'esame è possibile utilizzare materiale didattico (note, slide, libri, ecc.), in forma cartacea o elettronica. **NON è possibile comunicare con qualsiasi mezzo con altri che non siano i docenti.** Chi tenti di farlo vedrà annullata la sua prova.
- Nel caso in cui lo studente decida di utilizzare il computer, lo schermo deve essere condiviso attraverso Zoom durante tutta la durata della prova. **NON è possibile utilizzare la connessione a Internet ad eccezione della condivisione schermo con Zoom.**
- **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)

L'app store di un tipo di smartphone può gestire un catalogo di massimo 5000 app. Di queste, fino a 50 possono essere installate nello smartphone, a patto che la somma della memoria delle app installate sia minore o uguale a 3 GByte, che è la quantità di memoria riservata per l'installazione delle app. Le informazioni sulle app disponibili e quelle installate sono memorizzate nei due campi della struct `store` di tipo `app_store` riportata qui sotto, chiamati `app_esistenti` e `app_installate`. Il numero effettivo di app presenti nello store è rappresentato dal campo `num_app_esistenti`, mentre il numero di app installate sullo specifico dispositivo è rappresentato dal campo `num_app_installate`. I due vettori `app_esistenti` e `app_installate` sono riempiti di dati significativi, rispettivamente, fino alle celle di indice `num_app_esistenti-1` e `num_app_installate-1`. I restanti elementi sono da considerarsi non inizializzati. Come si può inferire dall'organizzazione del tipo struct `app`, per ogni app, viene salvato il nome, il prezzo, un punteggio (non negativo) calcolato sulla base delle review degli utenti, e la memoria richiesta per l'installazione. A partire dalle definizioni di tipi sotto riportati e dalla dichiarazione della variabile store, che si suppone già riempita, si risponda alle domande riportate seguenti.

```
#define MAX_APP_ESISTENTI 5000
#define MAX_APP_INSTALLABILI 50
#define MAX_MEMORIA_RISERVATA_APP 3*1024*1024*1024 /* 3 Gigabyte */

typedef struct{
      char nome[50];
      float prezzo;
      float punteggio;
      unsigned int memoria_richiesta;
} app;

typedef struct {
app app_esistenti[MAX_APP_ESISTENTI];
app app_installate[MAX_APP_INSTALLABILI];
int num_app_esistenti;
int num_app_installate;
} app_store;

app_store store;
```

1) (**2 punti**) Si scriva un frammento di codice in linguaggio C che calcoli e stampi a video quanto ha speso l'utente per acquistare tutte le app attualmente installate.

2) (**3 punti**) Si scriva un frammento di codice in linguaggio C che cerchi tra le app disponibili nello store quelle gratuite e, se ce ne sono, stampi a video il nome di tutte quelle con il punteggio più alto.

3) (**5 punti**) Si scriva un frammento di codice in linguaggio C che chieda all'utente il nome dell'app da installare, la cerchi nello store, stampi se può essere installata e, in caso affermativo, la aggiunga all'elenco delle app installate, aggiornando opportunamente la variabile store. Suggerimento: una app può essere installata se non è già installata, se c'è memoria sufficiente e se non si è raggiunto il numero massimo di app installabili.

Soluzione punto 1:

```
int i;
float tot_spesa = 0.0;
for(i = 0; i < store.num_app_installate; i++)
{
    tot_spesa += store.app_installate[i].prezzo;
}
printf("L'utente ha speso %.2f per installare le app\n", tot_spesa);
```

Soluzione punto 2:

```
int i;
float max = -1.0;
for(i = 0; i < store.num_app_esistenti; i++)
{
    if(store.app_esistenti[i].prezzo == 0 &&
       store.app_esistenti[i].punteggio > max)
            max = store.app_esistenti[i].punteggio;
    }
}
if(max != -1.0)
{
    for(i = 0; i < store.num_app_esistenti; i++)
    {
        if(store.app_esistenti[i].prezzo == 0 &&
           store.app_esistenti[i].punteggio == max)
            printf("%s\n", store.app_esistenti[i].nome);
    }
}
```

Soluzione punto 3:

```
int i, indice = -1, installata = 0;
unsigned int memoria;
char nome[30];
printf("Inserisci il nome dell'app da installare\n");
fgets(nome, 30, stdin);
for(i = 0; i < store.num_app_esitenti && indice == -1; i++)
{
    if(strcmp(nome, store.app_esistenti[i].nome) == 0) indice = i;
}
if(indice >= 0)
{
    for(i = 0; i < store.num_app_installate && installata == 0; i++)
    {
        if(strcmp(nome, store.app_installate[i].nome) == 0) installata = 1;
    }
    if(installata == 0)
    {
        if(store.num_app_installate < MAX_APP_INSTALLABILI)
        {
            memoria = store.app_esistenti[indice].memoria_richiesta;
            for(i = 0; i < store.num_app_installate; i++)
```

```
            {
                memoria += store.app_installate[i].memoria_richiesta;
            }
            if(memoria <= MAX_MEMORIA_RISERVATA_APP)
            {
                printf("L'app può essere installata\n");
                store.app_installate[store.num_app_installate] =
                    store.app_esistenti[indice];
               store.num_app_installate++;
            } else printf("Errore: memoria insufficiente\n");
        } else printf("Errore: massimo numero di app già installate\n");
    } else printf("Errore: l'app richiesta è già installata\n");
} else printf("Errore: l'app richiesta non esiste nello store\n");
```

Esercizio 2 (10 punti)
Un impianto industriale produce componenti meccanici per applicazioni aerospaziali. I componenti prodotti sono caratterizzati da un diametro esterno, un diametro interno e una lunghezza, e vengono prodotti in gruppi da 500 pezzi, che vengono misurati individualmente durante il processo produttivo con lo scopo di controllarne la qualità. Assumendo di partire da un workspace Matlab in cui sono già presenti tre array, `diametro_esterno`, `diametro_interno`, `lunghezza`, ognuno da 500 elementi, contenenti le misure di un gruppo di componenti, si risponda alle seguenti domande:

1) (**2 punti**) Si dichiarino tre vettori di tipo logical (logico) i cui elementi devono valere 1 per tutti e soli i componenti con, rispettivamente, un diametro esterno compreso tra 19.98 e 20.02, un diametro interno compreso tra 9.98 e 10.02, e una lunghezza compresa tra 29.9 e 30.1. Si dichiari infine un quarto vettore logical (logico) di nome `corretti` contenente un 1 per tutti e soli i componenti che rispettino contemporaneamente le tre precedenti condizioni.

2) (**2 punti**) Si memorizzino in un vettore `rapporto_preciso` gli indici degli elementi con un rapporto tra il diametro esterno e quello interno compreso tra 1.999 e 2.001.

3) (**3 punti**) Si stampino gli indici dei pezzi **non** corretti (che non rispettano cioè le tre condizioni indicate al punto 1), ma con un rapporto tra i diametri preciso (che rispettano cioè la condizione al punto 2).

4) (**3 punti**) Si disegni un grafico che riporti come ordinata il diametro interno dei soli pezzi **non** corretti. Si disegnino due linee orizzontali rosse ai limiti di accettabilità 9.98 e 10.02. Si aggiungano le etichette agli assi del grafico.

Soluzione punto 1:

```
v1 = diametro_esterno >= 19.98 & diametro_esterno <= 20.02;
v2 = diametro_interno >=  9.98 & diametro_interno <= 10.02;
v3 = lunghezza        >= 29.9  & lunghezza        <= 30.1;
corretti = v1 & v2 & v3;
```

Soluzione punto 2:

```
rapporto_diametri = diametro_esterno ./ diametro_interno;
rapporto_preciso = find(rapporto_diametri >= 1.999 & rapporto_diametri <= 2.001)
```

Soluzione punto 3:

```
for i = rapporto_preciso
    if ~corretti(i)
        disp(i);
    end
end
```

Soluzione punto 4:

```
y = diametro_interno(~corretti);
x = 1:length(y);
plot(x, y);
hold on;
plot(x,  9.98*ones(size(x)), 'r');
plot(x, 10.02*ones(size(x)), 'r');
xlabel('Pezzo');
ylabel('Diametro interno');
hold off;
```

Esercizio 3 (6 punti)

Per ognuna delle espressioni logiche riportate nelle righe della tabella, assumendo le seguenti dichiarazioni:

```
int x = 4, y = 9;
char c='q';
```

indicare se l'espressione è vera o falsa (scrivere V o F nella seconda colonna). Indicare inoltre, nella terza colonna, se l'espressione è vera per qualsiasi valore delle variabili (scrivere SI o NO) e, nella quarta colonna, se l'espressione è falsa per qualsiasi valore delle variabili (scrivere SI o NO). Si giustifichino (con calcoli, controesempi e/o ragionamenti) le risposte. Risposte prive di giustificazione non saranno prese in considerazione.

	Espressione	Vera o Falsa?	Sempre Vera?	Sempre Falsa?
1	`(y<5 && x>4) && (c>'a' || c<'z')`			
2	`(x>0) && (x==-x)`			
3	`(-y<=0) || (y>=0)`			
4	`(y>=10 || c=='c') || (c>='d' || c<'c')`			

Soluzione

Espressione	Vera o Falsa?	Sempre Vera?	Sempre Falsa?
`(y<5 && x>4) && (c>'a' || c<'z' )`	F	NO	NO
`(x>0) && (x==-x)`	F	NO	Sì
`(-y<=0) || (y >= 0)`	V	NO	NO
`(y>=10 || c=='c') || (c>='d' || c<'c' )`	V	Sì	NO

1) Inserendo i valori delle variabili abbiamo: (F && F) && (V || V) = F && V = F. Questo implica che non può essere sempre vera. Un controesempio per cui non è falsa è y=1, x= 5, c = 'b'.

2) Inserendo i valori delle variabili abbiamo (V) && (F) = F. Questo implica che non può essere sempre vera. È sempre falsa in quanto, per valori x>0 la prima parte dell'espressione risulta vera, ma la seconda è falsa. Per valori x≤0 invece la prima parte dell'espressione è falsa. Perché sia vera l'espressione devono essere vere entrambe le parti, quindi non lo sarà mai.

3) Inserendo i valori delle variabili abbiamo V || V = V. Questo implica che non può essere sempre falsa. Non è sempre vera, per esempio usando y=-5 l'espressione risulta falsa.

4) Inserendo i valori delle variabili abbiamo (F || F) || (V || F) = F || V = V. Questo implica che non può essere sempre falsa. L'espressione è sempre vera in quanto la seconda parte dell'espressione ha due elementi che sono V e F se c è un carattere successivo o uguale a 'd' o F e V se la variabile c è un carattere precedente a 'd'. Questo rende la seconda parte dell'espressione sempre vera, che in OR con la prima parte risulta sempre vera.

Domanda 1 (3 punti)

Spiegare, anche con un esempio diverso da quello presente nelle slide, come convertire un numero intero negativo in codifica complemento a 2.

Domanda 2 (3 punti)

Si presentino le differenze nella rappresentazione delle matrici nei linguaggi C e Matlab utilizzando dei semplici esempi per aiutare la spiegazione.