	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 18 Febbraio 2016		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene **3 esercizi** e deve essere debitamente compilato con cognome e nome, numero di matricola e posizione durante lo scritto (comunicata dal docente).
- 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) assicurandosi comunque che **quanto scritto sia ben leggibile**.
- È **vietato** utilizzare **calcolatrici, telefoni, tablet o pc**. Chi tenti di farlo vedrà **annullata** la sua prova.
- È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare. **Non è tuttavia possibile consultare temi d'esame degli anni precedenti**.
- 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.

Esercizio 1 (4 punti)

Un programma deve controllare se un numero **N** inserito dall'utente sia contemporaneamente:

- primo,
- dispari,
- compreso tra 3 e 100, estremi inclusi.

Quando il numero **N** inserito non soddisfa tutte le condizioni citate, l'inserimento deve essere ripetuto.

Per realizzare tale programma, un programmatore ha scritto il seguente frammento di codice Matlab, sfruttando la funzione **primo** che restituisce *true* se il suo argomento è un numero primo, *false* altrimenti:

```
while(primo(N) && (3 <= N) && (N <= 100) && (mod(N,2)))
 N = input('Inserire un numero: ');
end
```

Si risponda alle seguenti domande:

1. Si dica se il frammento di codice soddisfa i requisiti del programma. Nel caso in cui si ritenga che il frammento di codice non sia corretto, se ne indichi la motivazione e se ne fornisca un controesempio (ovvero un valore di **N** che ne mostri la non correttezza) spiegandolo.
2. Nel caso in cui si ritenga che il frammento di codice non sia corretto, se ne fornisca una versione in grado di soddisfare i requisiti indicati.
3. Si indichi se, e nel caso come, sia possibile semplificare la condizione riportata nel ciclo while per evitare ridondanze.
4. Si mostri la correttezza, o non correttezza, della condizione riportata nel ciclo while compilando una tabella di verità, ove:

   `A: primo(N)`

   `B: (3 <= N)`

   `C: (N <= 100)`

   `D: (mod(N,2)`

Soluzione

1. Un ciclo while viene ripetuto fino a quando la condizione che viene valutata e' vera! Requisito del programma è di TERMINARE l'inserimento (quindi avere la condizione == 0 del ciclo while) quando sono vere tutte le tre condizioni da verificare sul valore di N. La condizione che regola il ciclo while nel frammento di codice restituisce esattamente l'opposto. Pertanto il frammento di codice non è corretto.
2. La condizione corretta del ciclo while richiesto è esattamente la negazione di quella riportata nel frattempo di codice. Un controesempio è il numero N = 5, per il quale il ciclo viene eseguito nuovamente nonostante 5 sia numero primo, dispari e compreso tra 3 e 100.
3. La condizione (3 <= N) implica che il numero N sia maggiore di 3, la condizione primo(N) richiede che il numero sia primo, la condizione (mod(N,2)) e' invece inutile; infatti un numero primo e' sicuramente dispari, a meno que non sia pari a 2, ma questo valore viene escluso dalla condizione (3 <= N).

   L'espressione da valutare e' quindi: !(primo(N)&&(3 <= N)&&(N <= 100)).

   La sua tabella di verità è:

   A: primo(N)

   B: (3 <= N)

   C: (N <= 100)

   D: (mod(N,2))

A B C	(A && B)	(A && B && C)	!(A && B && C)
0 0 0	0	0	1
0 0 1	0	0	1
0 1 0	0	0	1
0 1 1	0	0	1
1 0 0	0	0	1
1 0 1	0	0	1
1 1 0	1	0	1
1 1 1	1	1	0

4.

```
A: primo(N)
B: (3 <= N)
C: (N <= 100)
D: (mod(N,2)
```

A	B	C	D	(B && C)	(B && C && D)	(A && B && C && D)
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	1

Esercizio 2 (10 punti)

Si scriva un programma in linguaggio C che permetta l'analisi delle foto delle montagne innevate caricate sul social network "*LibroFaccia*". Si considerino date le seguenti definizioni:

```
typedef char Stringa[30];
typedef struct{
    Stringa nome;
    Stringa cognome;
    Stringa username;
    int annoIscrizione;
} Utente;
```

che rappresentano, rispettivamente, una stringa all'interno di *LibroFaccia* e un utente del social.

Si svolgano i seguenti punti:

1. Si definisca un tipo di dato *Foto* atto a rappresentare una foto di un utente. Ogni foto è caratterizzata da un'immagine (una matrice di interi di dimensione MAX_DIM x MAX_DIM), da un titolo e da un tag (entrambi rappresentabili con una stringa di 30 caratteri).
2. Si definisca la costante MAX_DIM pari a 512. Come indicato al punto precedente, questa rappresenta il numero massimo di righe e colonne di un'immagine.
3. Si modifichi la definizione di Utente affinché sia in grado di
    a. contenere fino a 50 *Foto* per ogni utente (un utente potrebbe avere meno di 50 foto e in numero diverso dagli altri utenti);
    b. riportare quante foto siano contenute per ogni utente.
4. Si definisca una variabile *libroFaccia* contenente MAX_UTENTI variabili di tipo *Utente*. Si definisca inoltre la costante MAX_UTENTI con valore pari a 1000.

Si assuma poi che la variabile *libroFaccia* sia stata popolata interamente e che ogni utente sia stato popolato in tutti i suoi campi (comprese le proprie foto). Si scriva una porzione di codice in linguaggio C che:

5. Richieda all'utente di inserire un valore soglia *T*, controllando che sia compreso tra 0 e 255 (estremi inclusi).
6. Tra tutte le foto di tutti degli utenti di *LibroFaccia*, stampi a schermo il titolo della foto la cui matrice immagine ha il maggior numero di elementi che sono maggiori o uguali alla soglia *T* e che ha come tag "Montagna" o "Neve". Nel caso esistano più foto con tali caratteristiche, è sufficiente considerarne una qualsiasi di esse. Si stampi inoltre a schermo il nome e il cognome dell'utente che possiede tale foto.

Soluzione

1.

```
typedef struct{
    int immagine[MAX_DIM] [MAX_DIM];
    Stringa titolo;
    Stringa tag;
} Foto;
```

2.

```
#define MAX_DIM 512
```

3.

```
typedef char Stringa[30];
typedef struct{
    Stringa nome;
    Stringa cognome;
    Stringa username;
    int annoIscrizione;
    Foto fotoUtente[50];
    int nFoto;
} Utente;
```

4.

```
#define MAX_UTENTI 1000
Utente libroFaccia[MAX_UTENTI];
```

5.

```
int T;
do{
    printf("Inserire un valore soglia tra 0 e 255");
    scanf("%d", &T);
}while(T < 0 || T > 255)
```

6.

```
(#include <string.h>)

int i, j, k, h, num_utente, num_foto;
int sum_soglia = 0;
int max_sum_soglia = 0;
Stringa foto_trovata;
Stringa nome_trovato;
Stringa cognome_trovato;
for (i = 0; i < MAX_UTENTI; i++)
    for (j = 0; j < libroFaccia[i].nFoto; j++)
     if(strcmp(libroFaccia[i].fotoUtente[j].tag,"Montagna") == 0
     ||  strcmp(libroFaccia[i].fotoUtente[j].tag,"Neve") == 0)  {
          sum_soglia = 0;
          for (k = 0; k < MAX_DIM; k++)
              for (h = 0; h < MAX_DIM; h++)
                   if (libroFaccia[i].fotoUtente[j].immagine[k][h] >= soglia)
                        sum_soglia++;
          if (sum_soglia > max_sum_soglia) {
              max_sum_soglia = sum_soglia:
              num_utente = j;
              num_foto = i;
          }
     }

if (max_sum_soglia > 0)  {
   strcpy(foto_trovata, libroFaccia[num_utente].fotoUtente[num_foto].titolo);
   strcpy(nome_trovato, libroFaccia[num_utente].nome);
   strcpy(cognome_trovato, libroFaccia[num_utente].cognome);

   printf("Titolo foto trovata: %s\n", foto_trovata);
```

```
    printf("Scattata da: %s %s\n", nome_trovato, cognome_trovato);
}
else
    printf("Nessuna foto trovata");
```

Esercizio 3 (10 punti)

Si vuole visualizzare una stringa text su un display contenente **n_char** caratteri. Si assuma che **text** contenga più di **n_char** caratteri e che quindi il testo venga fatto scorrere verso sinistra fino a visualizzare una sola volta (ovvero non ciclicamente) tutti i caratteri contenuti in text, ad esempio come nei display che indicano la prossima fermata all'interno dei treni.

Per esempio, dato il testo "*buon viaggio*" e un display di 10 caratteri, nel display viene visualizzata la seguente sequenza di stringhe:

```
buon viagg
uon viaggi
on viaggio
n viaggio-
 viaggio--
viaggio---
iaggio----
aggio-----
ggio------
gio-------
io--------
o---------
----------
```

Si risponda alle seguenti domande:

- Si implementi in linguaggio Matlab un funzione **showRollingTxt** per visualizzare una stringa **text** su un display contenente **n_char** caratteri, come descritto sopra.

  In particolare la funzione **showRollingTxt** deve operare come segue:
  - Inizialmente vengono visualizzati (allineati a sinistra nel display) i primi **n_char** caratteri di **text**. Se text contiene meno di **n_char** caratteri, vengono introdotti a destra caratteri '-' (trattino) fino a riempire il display.
  - Il testo quindi "*scorre"* verso sinistra di un carattere, visualizzando **n_char** caratteri a partire dal secondo carattere di **text**; come nel caso precedente, se la parte di testo da visualizzare è più corta di **n_char** caratteri, si introducono caratteri '-'.

- La procedura descritta sopra continua fino a quando il display contiene **n_char** caratteri uguali a '-'.

Ogni volta che il testo scorre di un carattere, la funzione **showRollingTxt** stampa a schermo, su una riga diversa, il testo visualizzato nel display.

- Si scriva un script Matlab che invochi la funzione **showRollingTxt** per mostrare il testo "buon viaggio" su un display avente 10 caratteri.

Si implementi la funzione ricorsiva **showRollingTxtRic** che produce lo stesso risultato di **showRollingTxt** prendendo gli stessi parametri in ingresso ed uscita

Soluzione

```
function [] = showRollingTxt(text, n_char)
lenTXT = length(text);
txtToShow(1 : lenTXT + n_char) = '-';
txtToShow(1: lenTXT) = text;

for ii = 1 : lenTXT + 1
   disp(txtToShow(1 : n_char));
   % txtToShow(end + 1) = txtToShow(1); % per far continuare
   txtToShow(1) = [];
end



clear
clc
close all

t = 'buon viaggio';
n = 10;
showRollingTxt(t, n);
```

```
%Versione ricorsiva
function [] = showRollingTxtRic(text, n_char)

lenTXT = length(text);
txtToShow(1 : lenTXT + n_char) = '-';
if lenTXT == 0
   disp(txtToShow);
else
   txtToShow(1: lenTXT) = text;
   disp(txtToShow(1 : n_char));
   showRollingTxtRic(text(2:end), n_char);
end
```