	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **INFORMATICA B** 29 gennaio 2018		COGNOME E NOME
	Fila	Colonna	MATRICOLA
TEMA B			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 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!).
- **È vietato utilizzare calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- **È vietata la consultazione di libri, appunti e qualsiasi altra risorsa**.
- **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)

Si consideri il sistema di gestione dei treni in una stazione. In ogni stazione ci sono al più N_BANCHINE e ogni banchina consente la sosta e il transito di un treno. Ad ogni treno in stazione viene assegnato uno dei seguenti stati:

- *`fuoriStazione`*: il treno, in attesa di essere assegnato a una banchina, si trova fuori dalla stazione
- *`inIngresso`*: il treno è stato assegnato a una banchina ed è in marcia a velocità ridotta per entrare in stazione
- *`inSosta`*: il treno è fermo a una banchina
- *`attesaOut`*: il treno è fermo a una banchina in attesa di poter partire
- *`inUscita`*: il treno sta abbandonando la banchina procedendo a velocità ridotta.

Si assuma che siano già stati introdotti i tipi *Stazione* e *Banchina* (si veda codice qui sotto). In particolare, *Stazione* contiene l'insieme e il numero delle banchine presenti in stazione (*banchine*), la coda dei treni (al più 20) in attesa all'ingresso (*codaTreni*, *nTreniInCoda*) e una variabile Booleana che dice se c'e' qualche treno in manovra in stazione (*bloccata*). *Banchina*, invece, contiene il numero della banchina e i dati del treno in sosta, se la banchina non è in stato *libero*. Ciascun treno, oltre ad avere il proprio stato e il numero della banchina eventualmente assegnata, ha un campo *minutiAttesaOut* che indica da quanti minuti è in attesa in stato *attesaOut*.

```
#define N_BANCHINE 10
#define N_TRENI_CODA 20

typedef char stringa[20];
typedef enum { falso, vero } boolean;
typedef enum { libero, occupato } statoBanchina;
typedef enum
         { fuoriStazione, inIngresso, inSosta, attesaOut, inUscita } statoTreno;

typedef struct {
    stringa nome;
    statoTreno stTreno;
    int nBanchinaAssegnata;
    int minutiAttesaOut;
} Treno;

typedef struct {
  int numero;
  statoBanchina stBanchina;
  Treno trenoSosta;
} Banchina;

typedef struct {
  Banchina banchine[N_BANCHINE];
  int nBanchine; /* indica il numero di banchine presenti nella stazione */
```

```
  Treno codaTreni[N_TRENI_CODA];
  int nTreniInCoda;
  boolean bloccata;
} Stazione;
```

Supponendo che sia stata definita la variabile *stazioneMI* di tipo *Stazione* e che questa variabile sia stata riempita in modo opportuno per rappresentare lo stato corrente della stazione di Milano, si risponda alle seguenti richieste:

1. Dichiarando tutte le variabili necessarie, si definisca un frammento di programma che determina e stampa l'indice (nell'array di banchine) della banchina in cui è in sosta il treno che è in attesa da più tempo di poter partire.

2. Si definisca un frammento di programma che, se la stazione non è bloccata dalle manovre di qualche treno e se vi è almeno un treno in coda per entrare in stazione, blocca la stazione, estrae dalla coda il primo treno che vi era stato inserito e gli assegna lo stato *`inIngresso`*, e aggiorna la coda in modo tale che poi contenga solo i rimanenti treni in coda.

Soluzione

```
Stazione stazioneMI;

int i;
```

```
1.
boolean trenoInAttesa = falso;
int maxAttesaTreno = 0;
int indiceMaxAttesaTreno = 0;

for (i = 0; i < stazioneMI.nBanchine; i++) {
  Banchina p = stazioneMI.banchine[i];
  if (p.stBanchina == occupato && p.trenoSosta.stTreno == attesaOut) {
    trenoInAttesa = vero;
    if (maxAttesaTreno < p.trenoSosta.minutiAttesaOut) {
      indiceMaxAttesaTreno = i;
      maxAttesaTreno = p.trenoSosta.minutiAttesaOut;
    }
  }
}
if (trenoInAttesa) {
  printf("Treno con priorita' in attesa alla banchina n. %d",
         indiceMaxAttesaTreno);
}
```

```
    2.
    Treno trenoInManovra;
int j;

    if (stazioneMI.bloccata == falso && stazioneMI.nTreniInCoda > 0) {
      stazioneMI.bloccata = vero;
      trenoInManovra = stazioneMI.codaTreni[0];
      treniInManovra.stTreno = inIngresso;
      for(j = 0; j < stazioneMI.nTreniInCoda; j++) {
        stazioneMI.codaTreni[j] = stazioneMI.codaTreni[j + 1];
      }
      stazioneMI.nTreniInCoda--;
    }
```

Esercizio 2 (10 punti)

Si consideri il seguente problema: si vuole creare una matrice quadrata che sia organizzata come quella in figura.

2	2	2	2	2
2	3	3		2
2	3	4	3	2
2	3	3	3	2
2	2	2	2	2

1. Si scriva in linguaggio MATLAB una funzione iterativa **cornici** che, data la dimensione *N* della matrice e un numero di partenza *P*, restituisca al chiamante una matrice quadrata *NxN* così definita: la matrice contiene nella cornice più esterna il numero *P* e numeri crescenti nelle cornici più interne.

2. Si scriva inoltre uno script in linguaggio MATLAB che acquisisca da tastiera la dimensione desiderata *N* e il numero di partenza *P*, invochi la funzione **cornici** con gli opportuni parametri e infine stampi a video la matrice risultante.

3. Si implementi la funzione ricorsiva **corniciRic** che corrisponda alla versione ricorsiva della funzione **cornici**

Soluzione

1.

```
function [M] = cornici(N, P)
        % Questa funzione iterativa prende come parametro una dimensione e un numero
        % e restituisce al chiamante una matrice quadrata della dimensione data che
        % contiene nella prima riga, prima colonna, ultima riga e ultima colonna
        % il numero dato e nelle altre righe e colonne numeri crescenti in
        % modo da formare cornici concentriche.

        M = P * ones(N);
        ii = 1;
        while ii < N/2
      M(ii+1:N-ii, ii+1:N-ii) = P + ii;
      ii = ii + 1;
        end
      end
```

2.

```
% script
N = input('Inserisci la dimensione della matrice: ');
P = input('Inserisci il numero di partenza: ');
M = cornici(N, P)
```

3.

```
function [M] = cornici(N, P)
        % Questa funzione ricorsiva prende come parametro una dimensione e un numero
        % e restituisce al chiamante una matrice quadrata della dimensione data che
        % contiene nella prima riga, prima colonna, ultima riga e ultima colonna
        % il numero dato e nelle altre righe e colonne numeri crescenti in
        % modo da formare cornici concentriche.

        M = P * ones(N);
        if N > 2
      M(2:end-1, 2:end-1) = corniciRic(N-2, P+1);
        end
end
```

Esercizio 3 (6 punti)
Un cardiopatico ha uno smartwatch con 48 byte di memoria liberi e vorrebbe tenere in memoria l'ECG relativo alla sua attività cardiaca durante la notte. Il singolo dato è rappresentato dalla differenza di potenziale generata dal battito cardiaco (in Volt) ogni ora, e può essere un intero in intervallo [-1000 +2000] (estremi compresi). Si assuma che lo smartwatch utilizzi una codifica in Complemento a 2 e si risponda alle seguenti domande:

1. Quale è il numero B di bit necessari per registrare un'ora di attività cardiaca?

2. Quante ore è possibile registrare ancora nella memoria libera? Quanta memoria è necessaria per tenere in memoria 128 ore?

3. Supponiamo ora che siano state registrate le differenze di potenziale generate dal battito cardiaco delle prime 3 ore della notte, e che queste siano [135, -978, -1524]. Si codifichino questi numeri in Complemento a 2 con B bit e si calcoli la loro somma (in Complemento a 2), indicando se questa può essere registrata nello smartwatch, ovvero in B bit.

Soluzione

1. 2^10 = 1024 < 2000 < 2048 = 2^11 = 11 bit
   Con B = 12 bit si riesce a coprire l'intervallo [-2^11, 2^11 - 1] e quindi si riesce a registrare l'attività cardiaca di un'ora.

2. Essendo la memoria disponibile pari a 48 byte = 48 * 8 bit = 384 bit e poiché per ogni ora sono necessari 12 bit per tenere in memoria la rappresentazione della differenza di potenziale generata dal battito cardiaco, si possono tenere in memoria ancora al massimo 48 * 8 / 12 = 32 ore. Se quadruplicassi la memoria (i.e. 48 * 4 = 192 byte) otterrei spazio sufficiente per 32 * 4 = 128 ore

3. Tramite l'algoritmo delle divisioni successive $135_{10} = 10000111_2$, quindi il numero decimale 135_{10} in codifica Complemento a 2 con B = 12 bit è $135_{10} = 000010000111_{CP2}$.
   Dato che $988_{10} = 1111010010_2$, poichè -988 è negativo, devo cambiare il valore di tutti i bit ottenuti e sommare 1 per trasformare tale codifica in un numero negativo in Complemento a 2 con B = 12 bit, ovvero $-988_{10} = 110000101110_{CP2}$
   Analogamente $1524_{10} = 10101111100_2$ e il suo corrispettivo in Complemento a 2 con B = 12 bit è $-1524_{10} = 101010000100_{CP2}$
   La somma di tali tre valori esce dall'intervallo ammissibile con 12 bit, dunque non è possibile registrare la somma delle differenze di potenziale generate dal battito cardiaco delle prime 3 ore della notte nello smartwatch.

Esercizio 1 (10 punti)

Si consideri il sistema di gestione degli accessi delle grandi imbarcazioni a un porto. Nel porto ci sono *N* pontili di attracco (con *N* diverso per ciascun porto, ma mai maggiore di 10). Ogni pontile consente l'attracco di una nave.

Per garantire la sicurezza, solo una nave alla volta può manovrare nel porto. Per questo, a ogni nave che si trova nella zona del porto, viene assegnato uno dei seguenti stati:

1. *fuoriPorto*: in questo caso la nave ha richiesto l'ingresso in porto ma è ferma in attesa di un riscontro da parte dell'operatore portuale
2. *attesaIn*: l'operatore ha assegnato un pontile alla nave, ma ci sono altre navi in manovra e quindi la nave è in attesa del proprio turno
3. *manovra*: la nave sta facendo manovra nel porto
4. *attesaOut*: la nave ha richiesto l'autorizzazione a uscire dal porto ed è in attesa di ricevere l'ok all'inizio della manovra
5. *attraccata*: la nave è ferma al pontile.

Quando una nave chiede di entrare in porto, manda la richiesta all'operatore nella centrale di controllo e viene posta nello stato *fuoriPorto*. L'operatore verifica sul sistema di gestione se esiste un pontile disponibile per l'attracco. In caso negativo, chiede alla nave di attendere fuori dalla zona portuale, lasciando la nave in stato *fuoriPorto*. In caso affermativo, assegna alla nave il numero del pontile di attracco e, se ci sono altre navi in manovra, l'operatore mette la nave che ha effettuato la richiesta nella coda delle navi in attesa (al massimo 20) e le assegna lo stato *attesaIn*; altrimenti l'operatore mette la nave in stato *manovra* dandole così il via libera alle manovre, e blocca ogni nuovo ingresso/uscita al porto fino al completamento della manovra di attracco. Una volta che la manovra è stata completata, l'operatore pone la nave nello stato *attraccata*, sblocca gli ingressi/uscite dal porto e sceglie una nave in coda per eseguire un'altra manovra. Anche quando una nave deve uscire dal porto è necessario eseguire una procedura simile: la nave chiede il permesso all'operatore della centrale di controllo, se il porto è bloccato per un altro ingresso/uscita, l'operatore chiede alla nave di attendere assegnandole lo stato *attesaOut* e mettendola nella coda delle navi in attesa di ingresso/uscita; altrimenti le accorda il permesso di uscita, la mette nello stato *manovra* e libera il pontile.

Sono date le seguenti definizioni:

```
#define N_PONTILI 10
#define N_NAVI_CODA 20
typedef char stringa[20];
typedef enum {libero, occupato} statoPontile;
typedef enum {falso, vero} boolean;
typedef enum {attesaIn, attraccata, attesaOut, manovra, fuoriPorto} statoNave;

typedef struct {
  stringa nome;
  statoNave stNave;
  int nPontileAssegnato; /* il valore di questo campo è significativo se stNave è diverso da fuoriPorto */
} Nave;
```

```
typedef struct {
  int numero;
  statoPontile stPontile;
  Nave nave1; /* se statoPontile è libero, questo valore non è significativo */
} Pontile;

typedef struct {
  Pontile pontili[N_PONTILI];
  int nPontili; /* indica il numero di pontili presenti nel porto */
  Nave codaNavi[N_NAVI_CODA];
  int nNaviInCoda; /* indica il numero di navi in coda */
  boolean bloccato;
} Porto;
```

Supponendo che sia stata definita la variabile *portoNA* di tipo *Porto* e che questa variabile sia stata riempita in modo opportuno per rappresentare lo stato corrente del porto di Napoli, si risponda alle seguenti richieste:

1. Si definiscano tutte le variabili necessarie e si scriva un frammento di programma che:
   - acquisisce da tastiera il nome di una nuova nave che deve entrare in porto e pone lo stato della nave a *fuoriPorto*;
   - verifica se nel porto c'è un pontile per la nave e,
   - a seconda della situazione del porto, assegna il numero di un pontile alla nave e lo stato di occupato a tale pontile e gestisce l'assegnamento dello stato della nave ad *attesaIn* e il suo inserimento nella coda delle navi in attesa, oppure blocca il porto e pone lo stato della nave a *manovra*.
2. Definendo tutte le eventuali ulteriori variabili necessarie, si scriva un frammento di programma che, se il porto non è bloccato alle manovre e se vi è almeno una nave in coda, blocca il porto, estrae dalla coda delle navi in attesa la prima nave che vi è stata inserita, aggiorna tale coda, e assegna alla nave estratta dalla coda lo stato *manovra*, consentendole così di iniziare la manovra.

Soluzione

1.

```
Porto portoNA;
Nave nuovaNave;
int i;

/* acquisisce nome nuova nave e le assegna stato fuoriPorto */
printf("Inserire il nome della nave: ");
scanf("%s", nuovaNave.nome);
nuovaNave.stNave = fuoriPorto;
```

```
/* verifica se nel porto c'è un pontile per la nave */
	i = 0;
	while(i < portoNA.nPontili && portoNA.pontili[i].stPontile != libero)
		i++;

	if(i < portoNA.nPontili) /* c'e` posto per la nave */
	{
		nuovaNave.nPontileAssegnato = portoNA.pontili[i].numero;
		portoNA.pontili[i].stPontile = occupato;
		if(portoNA.bloccato == vero)  /* assegna stato nave ad attesaIn e suo inserimento in coda navi */
		{
			nuovaNave.stNave = attesaIn;
			portoNA.codaNavi[portoNA.nNaviInCoda] = nuovaNave;
			portoNA.nNaviInCoda++;
			printf("Accesso consentito, pontile numero %d, in coda d'attesa, aspettare il via \n",
					nuovaNave.nPontileAssegnato);
		}
		else /* blocca il porto e pone lo stato della nave a manovra */
		{
			portoNA.bloccato = vero;
			nuovaNave.stNave = manovra;
			printf("Accesso consentito, pontile numero %d, ingresso immediato \n",
					nuovaNave.nPontileAssegnato);
		}
	}
	else /* non c'e` un pontile per la nave */
		printf("Porto pieno, accesso vietato\n");


	2.

Nave naveInManovra;
	int j;

	if(portoNA.bloccato == falso && portoNA.nNaviInCoda > 0)
```

```
{
	/* e` possibile consentire a una nave in attesa di eseguire una manovra di attracco o uscita */
	portoNA.bloccato = vero;
	naveInManovra = portoNA.codaNavi[0]; /* estrae da coda navi in attesa la prima nave inserita */
	for(j = 0; j < portoNA.nNaviInCoda; j++)
		portoNA.codaNavi[j] = portoNA.codaNavi[j + 1];
	portoNA.nNaviInCoda = portoNA.nNaviInCoda - 1;
	naveInManovra.stNave = manovra;
	printf("Nave %s si prepari alla manovra\n", naveInManovra.nome);
}
```

Esercizio 2 (10 punti)

Si consideri il seguente problema: si vuole creare una matrice quadrata che sia organizzata come quella in figura.

10	10	10	10	10
10	9	9	9	10
10	9	8	9	10
10	9	9	9	10
10	10	10	10	10

1. Si scriva in linguaggio MATLAB una funzione iterativa **cornici** che, data la dimensione *N* della matrice e un numero di partenza *P*, restituisca al chiamante una matrice quadrata *NxN* così definita: la matrice contiene nella cornice più esterna il numero *P* e numeri decrescenti nelle cornici più interne.

2. Si scriva inoltre uno script in linguaggio MATLAB che acquisisca da tastiera la dimensione desiderata *N* e il numero di partenza *P*, invochi la funzione **cornici** con gli opportuni parametri e infine stampi a video la matrice risultante.

3. Si implementi la funzione ricorsiva **corniciRic** che corrisponda alla versione ricorsiva della funzione **cornici**

Soluzione

1.

```
function [M] = cornici(N, P)
    % Questa funzione iterativa prende come parametro una dimensione e un numero
    % e restituisce al chiamante una matrice quadrata della dimensione data che
    % contiene nella prima riga, prima colonna, ultima riga e ultima colonna
    % il numero dato e nelle righe e colonne successive numeri decrescenti in
    % modo da formare cornici concentriche.

    M = P * ones(N);
    ii = 1;
    while ii < N/2
        M(ii+1:N-ii, ii+1:N-ii) = P - ii;
        ii = ii + 1;
    end
end
```

2.

```
% script
N = input('Inserisci la dimensione della matrice: ');
P = input('Inserisci il numero di partenza: ');
M = cornici(N, P)
```

3.

```
function [M] = cornici(N, P)
        % Questa funzione ricorsiva prende come parametro una dimensione e un numero
        % e restituisce al chiamante una matrice quadrata della dimensione data che
        % contiene nella prima riga, prima colonna, ultima riga e ultima colonna
        % il numero dato e nelle righe e colonne successive numeri decrescenti in
        % modo da formare cornici concentriche.

        M = P * ones(N);
        if N > 2
      M(2:end-1, 2:end-1) = corniciRic(N-2, P-1);
        end
end
```

Esercizio 3 (6 punti)

Un corridore ha uno smartwatch con 44 byte di memoria liberi e vorrebbe tenere in memoria il bilancio calorico delle sue giornate di allenamento. Il bilancio calorico è rappresentato dalla differenza tra le kilo calorie assunte e quelle consumate, e può essere un intero in intervallo [-1000 +1000] (estremi compresi). Si assuma che lo smartwatch utilizzi una codifica in Complemento a 2 e si risponda alle seguenti domande:

1. Quale è il numero B di bit necessari per registrare un bilancio calorico giornaliero?

2. Quanti giorni è possibile registrare ancora nella memoria libera? Quanta memoria è necessaria per tenere in memoria 64 giorni?

3. Supponiamo ora che sia stato registrato il bilancio calorico dei primi 3 giorni, e che questo sia [245, -988, -324]. Si codifichino questi numeri in Complemento a 2 con B bit e si calcoli la loro somma (in Complemento a 2), indicando se questa può essere registrata nello smartwatch, ovvero in B bit.

Soluzione

1. 2^9 = 512 < 1000 < 1024 = 2^10 = 10 bit

Con B = 11 bit si riesce a coprire l'intervallo [-2^10, 2^10 - 1] e quindi si riesce a registrare il bilancio calorico di un giorno.

2. Essendo la memoria disponibile pari a 44 byte = 44 * 8 bit = 352 bit e poiché ogni giorno sono necessari 11 bit per tenere in memoria la rappresentazione della differenza tra calorie assunte e consumate, si possono tenere in memoria ancora al massimo 44 * 8 / 11 = 32 giorni. Se raddoppiassi la memoria (i.e. 44 * 2 = 88 byte) otterrei spazio sufficiente per 32 * 2 = 64 giorni.

3. Tramite l'algoritmo delle divisioni successive $245_{10} = 11110101_2$, quindi il numero decimale 245_{10} in codifica Complemento a 2 con B = 11 bit è $245_{10} = 00011110101_{CP2}$.

Dato che $988_{10} = 1111011100_2$, poichè -988 è negativo, devo cambiare il valore di tutti i bit ottenuti e sommare 1 per trasformare tale codifica in un numero negativo in Complemento a 2 con B = 11 bit, ovvero $-988_{10} = 10000100100_{CP2}$

Analogamente $324_{10} = 101000100_2$ e il suo corrispettivo in Complemento a 2 con B = 11 bit è $-324_{10} = 11010111100_{CP2}$

La somma di tali tre valori esce dall'intervallo ammissibile con 11 bit, dunque non è possibile registrare tale somma del bilancio calorico nello smartwatch.