	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-1; 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.

12	12	12	12	12
12	13	13	13	12
12	13	14	13	12
12	13	13	13	12
12	12	12	12	12

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.

Domanda teoria 1 (3 punti)

Quali sono le due parti fondamentali che non devono mai mancare in una funzione ricorsiva? A cosa servono?

Risposta

Una funzione ricorsiva deve essere sempre dotata di:

- chiamata ricorsiva condizionata: la funzione deve richiamare sé stessa almeno una volta; i parametri attuali con cui viene invocata la funzione devono cambiare e convergere verso il caso base.
- caso base: in cui la funzione non richiama se stessa; serve per terminare la sequenza di chiamate ricorsive e ritornare al chiamante una soluzione.

Domanda teoria 2 (3 punti)

Si consideri un calcolatore dotato di un sistema operativo multiutente e multiprogrammato e si consideri la seguente situazione: il calcolatore viene utilizzato simultaneamente da due utenti. Il primo utente sta usando un programma di chat (che di conseguenza richiede molti inserimenti da tastiera). Il secondo utente ha appena mandato in esecuzione una simulazione Matlab per il calcolo dell'aereodinamicità di un veicolo e ha già inserito tutti i parametri necessari al calcolo (la simulazione non richiede altri dati e visualizza i risultati solo al termine). Immaginando che, a questo punto, i due utenti lascino la loro postazione per andare a prendere il caffè, quale può essere la situazione dei processi attivi sul calcolatore al loro ritorno?

Risposta

Il primo utente ha mandato in esecuzione un programma di tipo reattivo che rimane in attesa di ricevere comandi per svolgere l'operazione richiesta. Questo programma quindi si troverà probabilmente in stato di attesa dopo che l'utente avrà lasciato la sua postazione. Il secondo utente ha lanciato un programma che si occupa dell'esecuzione di un calcolo. Questo potrà trovarsi nello stato di esecuzione, o di pronto (se nel frattempo qualche altro programma avrà preso il controllo), oppure potrà aver già terminato il suo compito.