	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Appello del 20 Luglio 2011		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, posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di un'ora e 30 minuti.
- Non separate questi fogli. Scrivete la soluzione **solo sui fogli distribuiti**. **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 propria prova.
- È ammessa la consultazione dei propri **libri** e **appunti**, purché con pacata discrezione e senza disturbare.
- 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 l'ammissione all'esame orale è necessario aver almeno impostato sufficientemente entrambi gli esercizi 1 e 2.
- La prova orale è parte integrante dell'esame e deve essere sufficiente per il superamento dell'esame complessivo.

Esercizio 1 (10 punti)

In presenza delle seguenti dichiarazioni

```
typedef struct { int p1, p2;  /*due numeri pari */
               } pari;
pari p;
```

Quesito 1: Scrivere un frammento di codice in **linguaggio C** che

1. legge da standard input un numero *n* intero positivo;
2. trova la coppia di numeri pari più vicini e minori o uguali al numero n, la memorizza in p e la stampa a video. Se per esempio, l'utente inserisce il valore 15, il frammento di codice inserirà in p i seguenti valori: p.p1: 14, p.p2: 12 e stamperà a video <14, 12>.

Quesito 2: In presenza della seguente dichiarazione di variabile:

```
pari arrayCoppiePari[100];
```

si scriva un nuovo frammento di codice (in linguaggio C) per fare in modo che, dato un valore *n* intero positivo letto da tastiera, il programma trovi, a partire da 0, le prime n coppie di numeri pari positivi, le memorizzi in *arrayCoppiePari* e le stampi a video. Quindi, per esempio, leggendo da tastiera il valore *n*=4, il frammento di codice dovrà memorizzare nell'array *arrayCoppiePari* e stampare a video i seguenti valori: <p1: 2, p2: 4>, <p1: 4, p2: 6>, <p1: 6, p2: 8>, <p1: 8, p2: 10>.
Si faccia in modo che il frammento di codice verifichi anche che il valore *n* sia positivo e compatibile con le dimensioni di *arrayCoppiePari*.

Soluzione Quesito 1

```
#include <stdio.h>

typedef struct { int p1, p2;  /*due numeri pari */
           } pari;


void main()
{
  pari p;
  int n;

   scanf("%d", &n);
```

```
  if (n > 3)
  {
      if (n % 2 != 0)
         p.p1 = n-1;
      else p.p1 = n;
      p.p2 = p.p1 - 2;
      printf("<%d, %d>\n", p.p1, p.p2);
  }
  else printf("%d <= 3\n", n);
}
```

Soluzione Quesito 2

```
#include <stdio.h>

typedef struct { int p1, p2;  /*due numeri pari */
            } pari;


void main()
{
   int n, cont, numero;
   pari arrayCoppiePari[100];

   scanf("%d", &n);
   cont = 0;
   numero = 2; /* primo numero pari */
   if (n > 0 && n < 100)
       for (cont=0; cont < n; cont++)
      {
         arrayCoppiePari[cont].p1 = numero;
         arrayCoppiePari[cont].p2 = numero + 2;
         numero = numero + 2;
         printf("<%d, %d>\n", arrayCoppiePari[cont].p1, arrayCoppiePari[cont].p2);
      }
}
```

Esercizio 2 (10 punti)

La *logistic map* è una sequenza di numeri reali x_0, x_1, x_2, ..., tali che, per ogni numero intero naturale n, si ha che:

$$x_{n+1} = r \cdot x_n \cdot (1 - x_n)$$

dove $0<x_0<1$ e r è un numero reale >0. Per esempio, prendendo $x_0=0.2$ e r=3.2, i primi quattro numeri della sequenza sono:

0.2000 0.5120 0.7995 0.5129

Scrivere la funzione con la seguente intestazione:

```
function [x, rip]=logisticMap(x0, r, lun)
```

che, ricevendo il primo valore x0 della sequenza, il coefficiente r, e un valore lun rappresentante la possibile lunghezza di sequenza, calcola in successione gli elementi della *logistic map*, fino a un numero massimo di lun elementi, e li inserisce nel vettore x. Se, prima di raggiungere il numero di elementi lun, si ottiene un elemento già generato in precedenza, allora il vettore x viene riempito con gli elementi fino a quello ripetuto (incluso) e il parametro rip viene posto a 1. Altrimenti, se nessun elemento si ripete, il vettore x viene riempito con i primi lun elementi della sequenza, e il parametro rip viene posto a 0.
Per esempio, con la chiamata:

```
>> [a b]=logisticMap(0.3, 2, 10)
```

si ottiene il risultato:

```
a =  0.3000    0.4200    0.4872    0.4997    0.5000    0.5000
b =  1
```

mentre con la chiamata:

```
>> [a b]=logisticMap(0.3, 1.3, 8)
```

si ottiene il risultato:

a = 0.3000 0.2730 0.2580 0.2489 0.2430 0.2391 0.2365 0.2348
b = 0

Soluzione

```
function [x, rip]=logisticMap(x0, r, lun)
x=x0;
while length(x)<lun
    x1 = r*x(length(x))*(1-x(length(x)));
    if any(x == x1)
        x(length(x)+1) = x1;
        rip=1;
        return;
    else
        x(length(x)+1) = x1;
    end;
end;
rip=0;
end
```

Esercizio 3 (6 punti)

Si consideri un sistema monoprocessore in cui si avviano tre diversi processi (A, B, C) che eseguono il seguente programma:

```
#include <stdio.h>
main {
  int a,b,s;
  printf("Inserire due numeri interi: ");
  scanf("%d %d", &a, &b);
  s = a + b;
  printf("La somma dei due numeri è: %d.\n",s);
}
```

Si consideri ciascuna delle situazioni elencate qui sotto e si dica se può verificarsi. Si giustifichi la risposta a ognuna delle 4 situazioni. Risposte senza giustificazione non saranno considerate nella determinazione del punteggio dell'esercizio.

1. A, B, C sono tutti e tre in stato di pronto.
2. A, B, C sono tutti e tre in stato di attesa.
3. A, B, C sono tutti e tre in stato di esecuzione.
4. A è in stato di attesa, B è in stato di esecuzione, C è in stato di pronto.

Risposta

1. Si può verificare se esiste un altro processo in stato di esecuzione.
2. Si può verificare. Accade quando tutti e tre i processi sono in attesa di input dell'utente.
3. Non si può verificare. In un sistema monoprocessore solo un processo alla volta può essere in stato di esecuzione.
4. Si può verificare. Accade ad esempio quando il processo A è in attesa di input dell'utente a causa dell'istruzione `scanf`, B sta eseguendo l'operazione di somma, C è pronto per essere eseguito.