	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 8 Novembre 2010		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene **4 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 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.
- 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)

Si consideri la seguente espressione booleana:

```
NOT (A AND NOT B) AND (NOT B OR C)
```

1. Si compili la seguente **tabella della verità** (in cui 0 rappresenta il valore logico FALSO, 1 il valore VERO)**:**

A	*B*	*C*	*NOT B*	*A AND NOT B*	*NOT (A AND NOT B)*	*NOT B OR C*	*NOT (A AND NOT B) AND (NOT B OR C)*
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

2. Si consideri ora la condizione, scritta in linguaggio C, in cui **x** e **y** siano due variabili int:

   !((x>3) && !(y>5)) && (!(y>5) || (x<2))

   ottenuta dalla prima formula sostituendo la variabile A con **x>3**, la variabile B con **y>5**, la variabile C con **x<2**

   Si risponda alle seguenti domande:
   a. L'espressione e' vera o falsa quando x=1 e y=7? (giustificare la risposta)
   b. Se y>5, per quali valori di x l'espressione e' vera? (giustificare la risposta)

Soluzione

1.

A	*B*	*C*	*NOT B*	*A AND NOT B*	*NOT (A AND NOT B)*	*NOT B OR C*	*NOT (A AND NOT B) AND (NOT B OR C)*
0	0	0	1	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	0	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	1	1	0	1	0
1	0	1	1	1	0	1	0
1	1	0	0	0	1	0	0
1	1	1	0	0	1	1	1

2.

Per x=1 e y=7, abbiamo A=falso, B=vero, C=vero per cui, dalla tabella della verità l'espressione risulta vera

Se y>5 allora B=vero. In questo caso, l'espressione è vera solo se C e' vera quindi solo per x<2.

Esercizio 2 (4 punti)

Il Triangolo di Tartaglia è una disposizione geometrica a forma di triangolo dei coefficienti binomiali. Un esempio di triangolo di Tartaglia di 5 righe è riportato qui sotto:

```
  1
 1  1
1  2  1
1  3  3  1
1  4  6  4  1
```

Il triangolo di Tartaglia può essere rappresentato in C con una matrice triangolare come mostrato qui sotto

1				
1	1			
1	2	1		
1	3	3	1	
1	4	6	4	1

Date le seguenti dichiarazioni:

```
#define N 10
int tartaglia[N][N];
```

si scriva il frammento di programma che, per ogni riga `r` (con `0 ≤ r < N`) della matrice `tartaglia`, verifichi che la somma degli elementi di tale riga sia pari a 2^r e stampi a schermo un messaggio contenente l'esito della verifica.

Per esempio, per la riga di indice 4 dovrà essere stampato il messaggio: "La somma di 1 + 4 + 6 + 4 + 1 è pari a 16" (se l'uguaglianza non fosse verificata, andrebbe invece visualizzato il messaggio: " La somma di 1 + 4 + 6 + 4 + 1 NON è pari a 16").

Note

- Non è necessario costruire o acquisire da tastiera la matrice `tartaglia`: si assuma che sia già stata inizializzata precedentemente.
- Per il calcolo delle potenze utilizzare la funzione di libreria `pow(a,b)` che calcola a^b

Soluzione

```
#include <stdio.h>
#include <math.h>
...
    int i, j, somma;

    /* qui va inserito il codice per l'assegnamento dei valori iniziali
       ad a tartaglia */.

    /* frammento di codice richiesto dall'esercizio */
    for(i=0;i<N;i++)
    {
        somma=1; printf("\n La somma di 1");
        for(j=1;j<=i;j++) {
            printf("+%d",tartaglia[i][j]);
            somma=somma+tartaglia[i][j];
        }
        if(somma != pow(2,i))
            printf(" NON ");
        printf("e' uguale a %d \n", pow(2,i));
    }
}
```

Esercizio 3 (5 punti)

La descrizione delle modalità d'esame di un corso universitario recita quanto segue.
"Durante il corso sono previste due prove scritte in itinere non obbligatorie: gli studenti possono partecipare, a loro scelta, a una o a entrambe. Se entrambe le prove sono valide e se la somma dei punteggi conseguiti è ≥ 18, lo studente ha superato l'esame del corso senza dover sostenere altre prove. Ogni prova in itinere assegna un massimo di 17 punti, e la prova in itinere è valida ai fini del superamento dell'esame solo se il voto è ≥ 8."

Con riferimento alle seguenti dichiarazioni di tipo e di variabile:

```
typedef char stringa[30];
typedef char matricola[10];
typedef struct { stringa cognome, nome;
                 matricola m;
               } datiStudente;

typedef struct { datiStudente  stud;
                 /* dati presenza e voto delle prove intermedie */
                 /* se la prova intermedia è sostenuta, presenza!=0, altrimenti ==0 */
                 int pres1, pres2;
                 int voto1, voto2;
               } datiProveStudente;

typedef struct { datiProveStudente s[300];
                 int nStud;
                 /* numero studenti effettivamente presenti nell'array.
                    I loro dati occupano le prime nStud posizioni di s */
               } registroProveInt;

registroProveInt r;
```

1. Assumendo che la variabile `r` sia stata già precedentemente inizializzata, si scriva un frammento di codice, dichiarando eventualmente opportune variabili aggiuntive, che stampi a schermo la matricola e i punti ottenuti dagli studenti che hanno partecipato a una sola delle due prove in itinere;

2. (**facoltativo**) Con riferimento alle seguenti ulteriori dichiarazioni di tipi e di variabili:

```
typedef struct { matricola m[300];
                 int punti [300];
                 int nStud; /* come sopra indica numero studenti */
               } registroEsiti;

registroEsiti  neg;
```

si scriva una variante del codice precedente che, invece di stampare a video matricole e punteggi, li inserisca nella variabile `neg`; si faccia in modo che, se gli studenti sono in numero N<300, i loro dati siano memorizzati, senza discontinuità, nella parte iniziale di lunghezza N dell'array, e che il campo `nStud` di `neg` sia uguale a N.

Soluzione

1.

```
int i;
for (i=0; i<r.nStud; i++)
  if ( !(r.s[i].pres1 && r.s[i].pres2) &&
       (r.s[i].pres1 || r.s[i].pres2) ) {
     printf("\n matricola %s ", r.s[i].stud.m);
     if (r.s[i].pres1 )
        printf(" punteggio %d", r.s[i].voto1);
     else
        printf(" punteggio %d", r.s[i].voto2);
  }
```

2.

```
int i;

neg.nStud=0;
for (i=0; i<r.nStud; i++)
  if ( !(r.s[i].pres1 && r.s[i].pres2) &&
       (r.s[i].pres1 || r.s[i].pres2) ) {
     strcpy(neg.m[neg.nStud], r.s[i].stud.m); /* neg.m[pos.nStud]=r.s[i].stud.m; */
     if (r.s[i].pres1 )
       neg.punti[neg.nStud]= r.s[i].voto1;
     else /* if r.s[i].pres2 */
       neg.punti[neg.nStud]= r.s[i].voto2;
     neg.nStud++;
  }
```

Esercizio 4 (4 punti)

1. Si determini la codifica del valore 16.0 secondo lo Standard IEEE 754-1985 a precisione singola, riportando i calcoli effettuati.
2. Si determini la codifica del valore 0.8 secondo lo stesso standard, sempre riportando i calcoli effettuati.
3. Si consideri il seguente programma C e si indichi l'effetto della sua esecuzione, motivando adeguatamente la risposta.

```
#include <stdio.h>
main() {
  float f;  int i;

  f=0.8;
  for (i=1; i<20; i++)
    f = f+0.8;
  printf("\nIl numero 0.8*20 ");
  if (f != 16.0) printf("non ");
  printf("e' uguale a %f", 16.0);
}
```

Soluzione

1.
$16_{10}=10000_2$, e inoltre la parte frazionaria del numero è nulla, quindi la forma normalizzata è $1.00000000000000000000000 \cdot 2^4$ e la mantissa è composta da 23 zeri, come sempre avviene per i numeri che sono uguali a una potenza di 2.
La caratteristica, in eccesso K, con K=127, è $c=4_{10}+127_{10}=131_{10}=10000011_2$. Quindi la rappresentazione di 16 è 0 10000011 00000000000000000000000. Si noti che la rappresentazione è esatta, nel senso che non sono state introdotte approssimazioni.

2.
Per il numero 0.8 si ottiene la rappresentazione della parte frazionaria con i seguenti calcoli

```
        0.8   X
           2
      ---------
1 <-    1.6
        0.6   X
           2
      ---------
1 <-    1.2
        0.2   X
           2
      ---------
0 <-    0.4
        0.4   X
           2
      ---------
0 <-    0.8
```

Poiché si è riottenuto il numero 0.8 di partenza, la rappresentazione in base 2 è periodica, $0.8_{10}=0.\underline{1100}_2$ (il periodo è mostrato sottolineato) e la rappresentazione in forma normalizzata è $1.\underline{1001}_2 \cdot 2^{-1}$. La mantissa è perciò 10011001100110011001100, la caratteristica è $c=-1_{10}+127_{10}=126_{10}=01111110_2$. Quindi la rappresentazione di 0.8 è 0 01111110 10011001100110011001100. Si noti che la rappresentazione è approssimata, per via del fatto che la parte frazionaria è periodica.

3.
Viene visualizzata la scritta

```
Il numero 0.8*20 non e' uguale a 16.0
```

a causa degli errori che si accumulano durante le operazioni di somma, poiché l'addendo è rappresentato in modo approssimato, e la somma viene confrontata col numero 16.0 che invece è rappresentato in modo esatto.