Politecnico di Milano
Dipartimento di Elettronica e Informazione
prof. Vincenzo Caglioti

Fondamenti di Informatica – Prova Intermedia 24 novembre 2016

Il punteggio massimo è di 30 punti e la prova contribuisce alla valutazione finale solo se il voto ottenuto è non inferiore a 18: in caso contrario occorrerà sostenere la prova in uno degli appelli previsti.

Matricola _____ **Cognome**_____ **Nome** _____

Istruzioni

- La prova dura 2 ore.
- Scrivere solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità e cancellando le parti di brutta con un tratto di penna. Non separare questi fogli. Si può utilizzare la matita purché non rossa.
- È vietato portare all'esame libri, eserciziari, appunti, calcolatrici, cellulari o tablet. Chiunque venga trovato in possesso di documentazione relativa al corso – anche se non strettamente attinente alle domande proposte – vedrà annullata la propria prova.
- È vietato comunicare con i colleghi. Rivolgere eventuali domande al docente presente.
- Non è consentito uscire dall'aula durante prima di consegnare la prova o lasciare l'aula conservando il tema della prova in corso.

Voti parziali degli esercizi e voto finale:

Esercizio 1 (Punti 6 + 4 =10)_____

Esercizio 2 (Punti 1 + 6 + 10 =17) _____

Esercizio 3 (Punti 3) _____

Voto finale: _____

Esercizio 1. Punti 10

Un array di 8 interi (con i soli due valori significativi 0 e 1) è utilizzato per contenere i bit che costituiscono la rappresentazione in complemento a due di un intero compreso tra -128 e +127.

A. Scrivere un programma che legge da tastiera un intero negativo, compreso tra -128 e -1, ne calcola la rappresentazione in complemento a due e ne memorizza i relativi bit in un array di 8 interi. **Pt 6**

B. Completare -con dichiarazioni e istruzioni- il seguente programma che, inizializzati due array di 8 interi, ciascuno contenente la rappresentazione in complemento a due di un intero compresi tra -127 e +128, (i) ne effettua l'addizione (somma algebrica), (ii) memorizza il risultato in un terzo array di 8 interi, (iii) stampa un avviso in caso di "overflow". **Pt 4**

```
#include <stdio.h>
int main()
{ … /* completare con dichiarazioni */
    int M[8] = {1, 0, 0, 1, ….};
    int N[8] = {0, 1, 1, 1, ….};
 …. /* completare con istruzioni */
}
```

Possibili soluzioni (in ***corsivo***)**:**

A.

```
#include <stdio.h>
int main()
{
    int A[8];
    int n;
    int q;
    int i;
    scanf("%d",  &n);
    q = -n;
    for (i=7; i>=0; i--)
    {
        A[i] = q % 2;
        q = q/2;
    }
    i = 7;
    while (A[i] == 0) i--;
    for(i--;i >= 0; i--) A[i] = 1 - A[i];
    return (0);
}
```

B.

```
#include <stdio.h>
int main()
{    int Somma[8];
     int i;
     int carry=0;

     int M[8] = {1, 0, 0, 1, ….};
     int N[8] = {0, 1, 1, 1, ….};
     for (i=7; i >= 0, i--)
     {
          Somma[i] = (carry + M[i] + N[i]) % 2;
          carry = (carry + M[i] + N[i]) / 2;
     }
     if (M[0] == N[0]) && (Somma[0] != M[0] * N[0]) printf("overflow");
     return (0);
}
```

Esercizio 2 (17 punti).

Si consideri una **matrice** di 10 x 10 celle quadrate, ciascuna delle quali può essere piena o vuota. Ogni **cella** è perciò rappresentata da un intero che può assumere solo il valore 1 o il valore 0, a indicare rispettivamente cella piena e cella vuota. Due celle si dicono ***vicine*** se esse si trovano in posizioni consecutive lungo una stessa riga oppure lungo una stessa colonna della matrice. La **posizione** di una cella è un aggregato contenente l'indice di riga I e l'indice di colonna J dell'elemento di matrice che rappresenta la cella stessa.

Un **oggetto** è l'insieme delle celle piene di una matrice; esso è rappresentato tramite una lista sequenziale di al più 100 posizioni. L'***area*** dell'oggetto presente nella matrice è data dal numero di celle piene in tale matrice. Il ***perimetro*** dell'oggetto è il numero di coppie di celle vicine tali che una delle due celle è piena mentre l'altra è vuota (si veda l'esempio riportato in Figura).

NOTA: si assuma che non esistano celle piene nella prima/ultima riga/colonna della matrice

Ad esempio l'area dell'oggetto visualizzato nella figura è pari a 18; il suo perimetro è pari a 38, pari cioè al numero di lati che si trovano al confine tra una cella piena e una cella vuota.

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	0	0	1	1	0	1	1	0
0	0	0	1	1	1	0	0	1	0
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0

Figura: una parte della matrice con alcune celle piene (in grigio); il perimetro è dato dalla lunghezza totale del confine in grassetto, ovvero dal numero di coppie di celle vicine e di diverso valore (colore).

A. Si definiscano i tipi di dato **matrice** e **oggetto**. **Punti 1**

B. Si completi con dichiarazioni e istruzioni il seguente programma che, inizializzata una **matrice** M, costruisca l'oggetto presente nella matrice, e ne stampi l'area. **Punti 6.**

```
#include <stdio.h>
int main()
{ … /* completare con dichiarazioni */
    int M[10][10] = {{1,0,0,1,1, ….},
                     {0,1,1,0,1, ….}
                      … …
                     {1,1,0,1,0, ….}}
 …. /* completare con istruzioni */
}
```

C. Si completi con dichiarazioni e istruzioni il seguente programma che, inizializzata una **matrice** M, calcoli il perimetro dell'oggetto presente nella matrice. **Punti 10.**

```
#include <stdio.h>
int main()
{ … /* completare con dichiarazioni */
    int M[10][10] = {{1,0,0,1,1, ….},
                     {0,1,1,0,1, ….}
                      … …
                     {1,1,0,1,0, … }}
 …. /* completare con istruzioni */
}
```

Possibili soluzioni *(*in *corsivo)*:

A.

```
typedef  int Matrice[100][100];
typedef struct {int I; int J;} Posizione;
typedef struct {int N; Posizione Seq[10000];} Oggetto;
```

B.

```
#include <stdio.h>
int main()
{    int i,j;
     Oggetto Obj;
     int M[10][10] = {{1,0,0,1,1, ….},
                      {0,1,1,0,1, ….}
                       … …
                      {1,1,0,1,0, … }}
     Obj.N = 0;
     for (i=0; i<100 - 1; i++)
          for (j=0; j<100 - 1; j++)
               if (M[i][j] == 1)
               {
                    Obj.Seq[Obj.N].I = i;
                    Obj.Seq[Obj.N].J = j;
                    Obj.N++;
               }
     printf("Area: %d", Obj.N);
}
```

C.

```
#include <stdio.h>
int main()
{ … int i,j;
     int perim;
     int M[10][10] = {{1,0,0,1,1, ….},
                      {0,1,1,0,1, ….}
                      … …
                      {1,1,0,1,0, … }}
     perim = 0;
     for (i=0; i<100 - 1; i++)
          for (j=0; j<100 - 1; j++)
          /* esplora solo a destra e in giù per non duplicare le coppie */
               {  if (M[i][j] + M[i][j+1] = = 1) perim++;
                  if (M[i][j] + M[i+1][j] = = 1) perim++;
               }
     printf ("Perimetro: %d", perim);
}
```

Esercizio 3 (Punti 3)
/* Si indichi che cosa stampa a schermo il programma seguente, dopo che esso è stato modificato inserendo, al posto di 999999, il numero di matricola dello studente. */

```
#include <stdio.h>
#define LUNG 6

int main()
{
   char Uscita[LUNG+1];
   char *P1, *P2;
   char Matricola[LUNG+1] = "999999";
   /* si ricorda che questo tipo di inizializzazione inserisce
    * automaticamente il carattere di terminazione */

   Uscita[LUNG] = Matricola[LUNG];
   P1 = Matricola;
   P2 = Uscita + LUNG - 1;

   while ('\0' != *P1)
   {
     *P2 = *P1;
     ++P1;
     --P2;
   }

   printf("%s", Uscita);

   return 0;
}
```