3.3 Soluzioni

Soluzione dell'esercizio 3.1

```
#include <stdio.h>

void main()
{
    int n, i, primo;

    do {
        printf("Inserire un intero: ");
        scanf("%d" , &n);
    } while(n < 1);

    /*
      Variabile di flag: il suo valore cambia da 1 a 0 quando trovo
         un
      divisore. Nell'inizializzazione della variabile, assumiamo che
      il numero inserito sia primo.
    */
    primo = 1;

    /*
      Occorre escludere 1 e n perche' dobbiamo considerare solo i
      divisori propri. Quindi iniziamo da 2.
     */
    i = 2;

    /*
      Inutile controllare i numeri da n/2 + 1 a n: non contengono
         divisori.

      Arresto il ciclo quando la variabile di flag primo diventa 0: è
      inutile cercare altri divisori
    */
    while (i <= n/2 && primo == 1) {
        if (n % i == 0) //se divisore trovato -> numero non primo!
            primo = 0;

        i++;
    }

    /*
      Qui, al termine del ciclo, se primo == 1 vuol dire che non si è
      mai verificato n % i == 0, quindi non esistono divisori propri
      ed n è primo.
    */
```

```
    printf("\n%d", n);

    if (primo == 0)
        printf(" non "); // il corpo dell'if è solo quest'istruzione
    printf(" è primo\n");

    getchar();
}
```

```
#include <stdio.h>

void main()
{
    int n, i, trovato;

    do {
        printf("Inserire un intero: ");
        scanf("%d" , &n);
    } while(n < 1);

    for (i = n/2; i > 1 && n % i != 0; i--);

    /* //In alternativa

       i = n/2;

       while (i > 1 && n % i != 0)
           i--;

     */

    if (i > 1)
        printf("Divisore %d trovato: numero non primo.", i);
    else //i == 1
        printf("Divisore non trovato: numero primo.");

    getchar();
}
```

Soluzione dell'esercizio 3.2

```
#include <stdio.h>

void main()
{
    int n = 0;
    int i;
```

```
int primo;
int nPrimi;
int j;

do {
    printf("Inserire il nr di primi:");
    scanf("%d", &nPrimi);
} while(n < nPrimi);

n = 1;
j = 1;

while (j <= nPrimi) {
    // qui inizia il blocco di codice per valutare se n  primo
    primo = 1;

    i = 2; // non devo controllare 1 perch uno divide tutti

    /*
      cos facendo quando n  1 non entro nel ciclo e concludo
      correttamente che 1  primo
    */
    while (i <= n/2 && primo == 1) {
        if(n % i == 0)
            primo = 0;
        i++;
    } // chiude ciclo per cercare i divisori

    /*
      se la variabile di flag primo  rimasta == 1, allora non
      mai capitato n % i == 0, quindi non esiste 2 <= i <= n /2
      che divide n
    */
    if(primo == 1)
    {
        printf(" %d " , n);

        /*
          incremento la variabile che conta il numero di primi
          incontrati
        */
        j++;
    }

    n++; // passo a valutare il prossimo intero,

    // indipendentemente dal fatto di aver trovato un numero
       primo o no.
```

```
    } // chiude ciclo per cercare nPrimi numeri primi

} // chiude main
```

Soluzione dell'esercizio 3.3

```
#include <stdio.h>

void main()
{
    int n, i;

    do {
        printf("Inserire un intero ");
        scanf("%d" , &n);
    } while(n < 1);

    // soluzione con il ciclo while
    /*
    i = 1; //init_expr

    while(i <= n) {
        if(n % i == 0)
            printf(" %d " , i);

        i++;
    }
    */

    // soluzione alternativa con il for,

    /*
      NB: l'init_expr i = 1; e la loop_expr i++; fanno parte della
      prima riga nel ciclo. Nel ciclo while erano prima del ciclo e
      all'interno rispettivamente.
    */
    for(i = 1; i <= n ; i++)
        if(n % i == 0)
            printf(" %d ", i);

}
```

Soluzione dell'esercizio 3.4

```
#include <stdio.h>

void main()
```

```
{
    int n, m, MCD, i;

    do {
        printf("Inserire n = ");
        scanf("%d" , &n);

        printf("Inserire m = ");
        scanf("%d" , &m);

    } while(n < 1 || m < 1);
    // contiuno a chiedere l'immissione se o m o n non  positivo

    // provo tutti i numeri minori o uguali ad n e ad m
    for (i = 1; i <= n && i <= m; i++)
        if(n % i == 0 && m % i == 0)
            MCD = i; //trovato, ma continuo a cercare

    printf("l'MCD e' %d\n" , MCD);
}
```

Soluzione dell'esercizio 3.5

```
#include <stdio.h>

void main()
{
    char g1, g2;

    /*
      variabile che indica il vincitore:
         - se vince == 1 allora vince g1
         - se vince == 2 allora vince g2
         - se vince == 0 allora parita'
     */
    int vince = 1;

    int ok1, ok2;

    do {
        printf("Giocata giocatore 1 (C/F/S?): ");
        scanf("%c", &g1);

        fflush(stdin);

        printf("\nGiocata giocatore 2 (C/F/S?): ");
        scanf("\n%c", &g2);

        fflush(stdin);
```

```
        // variabile ausiliaria, vale 1 se la giocata del giocatore 1
             buona
        ok1 = (g1 == 'C' || g1 == 'F' || g1 == 'S');

        // variabile ausiliaria, vale 1 se la giocata del giocatore 2
             buona
        ok2 = (g2 == 'C' || g2 == 'F' || g2 == 'S');

        /*
          la condizione di permanenza nel ciclo : "quello inserito da
          g1 non va bene o quello inserito da g2 non va bene"
        */
    } while(!ok1 || !ok2);

    /*
      per ridurre il numero di casi dell'if posso assumere che vinca
      sempre g1 e poi valuto le condizioni che farebbero vincere g2
     */

    // i caratteri nel codice vanno tra apici singoli ' '
    if(g1 == 'C' && g2 == 'F')
        vince = 2;

    if(g1 == 'F' && g2 == 'S')
        vince = 2;

    if(g1 == 'S' && g2 == 'C')
        vince = 2;

    // e quelle che farebbero pareggiare
    if(g1 == g2)
        vince = 0;

    // stampo il risultato
    if (vince > 0)
        printf("vince il giocatore %d\n" , vince);
    else
        printf("pari\n");

    getchar();
}
```