Esercizio 1

Scrivere un programma che acquisisca da tastiera una sequenza di caratteri terminata da '!' e riporti sul monitor una sequenza derivata dalla precedente secondo le regole dell'alfabeto *farfallino* (ogni vocale viene raddoppiata inserendovi in mezzo una *f*).

Esempio:

quanto mi piace questo corso!

q**ufuafa**nt**ofo** m**ifi** p**ifiafa**c**efe** q**ufuefe**st**ofo** c**ofo**rs**ofo**!

Per semplicità, si considerino solo le lettere minuscole.
Ogni carattere diverso dalle vocali deve restare inalterato.
Si acquisisca da tastiera un carattere alla volta.

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)
using namespace std;

int main( ) {
    char c;
    cout << "Inserisci una frase: \n";
    do {
        cin.get(c);
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')
          cout << c << "f" << c;
        else
          cout << c;
   } while (c != '!');
   cout << endl;

   system("PAUSE");
   return EXIT_SUCCESS;
} // end main
```

Estensioni e Varianti

- Regola del vamavvo: tutte le 'r' sino sostituite con delle 'v'
- Variante fiorentina: tutte le 'c' che siano iniziali di parola siano sostituite con 'h'
- Variante urlata: la frase sia trascritta interamente in lettere maiuscole
- Applicandole tutte insieme, ad esempio:
  che corso! → HHEFE HOFOVSOFO!

Esercizio 2

Dato un numero positivo Q,

- scrivere la sua rappresentazione in binario naturale, applicando il tradizionale algoritmo per divisioni successive
  (per convenzione, in questo esercizio l'output si intende corretto se letto da destra a sinistra);
- indicare anche il minimo numero di bit utilizzato.

Es:
Input: 19 in decimale, Output: 11001 in binario.

Stesura informale dell'algoritmo: (lasciata come esercizio)

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)
using namespace std;

int main( )
{
      int n; /* contatore */
      int Q; /* variabile per il numero letto da tastiera */
      /* ciclo di controllo per l'immissione dati che garantisce Q>0 */
      do {
            cout << "\n Inserisci un numero positivo Q: ";
            cin >> Q;
      } while (Q <= 0);
      cout << "\n In binario: ";
      n = 0;
      do {
            cout << (Q % 2);
            Q = Q / 2;
            n = n+1;
      } while (Q != 0);
      cout << "\n numero di bit n = " << n << endl;

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

ESERCIZIO 3

Si scriva un programma in linguaggio C che letto un numero intero positivo dallo standard input, visualizzi a terminale il quadrato del numero stesso facendo uso soltanto di operazioni di somma.
Si osservi che il quadrato di ogni numero intero positivo N può essere costruito sommando tra loro i primi N numeri dispari.
Esempio: N = 5; $N^2 = 1 + 3 + 5 + 7 + 9 = 25$.

Stesura informale dell'algoritmo:
Premessa:
l' idea di soluzione è quella di scandire i primi N numeri dispari esprimendoli nella forma (i+i+1) al variare dell'intero i tra 0 e N-1 e accumulare la loro somma man mano che si procede nella loro scansione in un'altra variabile.
N2 = (2·0+1) + ... + (2·i+1) + ... + (2·(N-1)+1);
quindi si utilizzeranno almeno 3 variabili: N, i, S rispettivamente per il numero, il contatore e l' accumulatore.

1. Inizio dell' algoritmo
2. Leggi un numero intero positivo dallo standard input.
3. Inizializza un contatore i a 0
4. Inizializza un accumulatore S a 0
5. Finché il valore del contatore è minore del numero letto
    a. Somma all' accumulatore il doppio del valore del contatore incrementato di 1
    b. Incrementa il contatore
    c. Torna al punto 5.
6. Stampa a terminale il valore dell' accumulatore
7. Fine dell' algoritmo

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main( )
{
      /* dichiarazione delle variabili */
      int i; /* contatore */
      int N; /* variabile di cui si vuole calcolare il quadrato */
      int S; /* accumulatore per il risultato del calcolo */

      /* ciclo di controllo che garantisce N > 0 */
      do
      {
            cout << "\n Inserisci un numero positivo N: ";
            cin >> N; /* legge N dallo standard input */
            /* Finché il numero inserito non è positivo ripetere */
            /* l'immissione dati. */
      } while (N <=0 );
      /* Il numero inserito è positivo */

      S = 0; /* inizializzazione della variabile di accumulo */

      /* ciclo di scansione dei primi N numeri dispari */
      i = 0;
      while(i < N)
      {
            /* Finché il contatore è minore del numero letto */
            /* aggiorna il contenuto della variabile accumulatore */

            S = S + (i+i+1);
            i = i + 1; /* incrementa il contatore */
      }

      cout << "\n Il quadrato del numero inserito e': " << S << "\n";

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

ESERCIZIO 4

Si definisce *Triangolare* un numero costituito dalla somma dei primi N numeri interi positivi per un certo N.
Esempio: per Q = 10 si ha Q = 1 + 2 + 3 + 4, da cui N=4.
Scrivere un programma C che stabilisca se un numero intero positivo Q, letto dallo standard input, è un numero triangolare o meno, utilizzando soltanto operazioni tra numeri interi.
In caso affermativo stampare a video il numero inserito e il massimo degli addendi che lo compongono.

Stesura informale dell'algoritmo

Idea di soluzione: se Q – 1 – 2 – 3 – ... – i – ... – N == 0 per un certo N allora Q è Triangolare.

1. leggi il numero positivo Q dallo standard input
2. inizializza un contatore i a zero;
3. memorizza in una variabile S il valore della variabile in ingresso.
4. Finché il numero S è maggiore di zero
    4.1. incrementa di 1 il valore del contatore
    4.2. sottrai a S il valore del contatore i
    4.3. torna a 4.
5. Se (il valore residuo di S è zero) allora
    5.1. il numero è triangolare
    5.2. il valore del massimo degli addendi è uguale al contatore i
    5.3. la variabile Q contiene il valore della variabile in ingresso
6. altrimenti il numero NON è triangolare.

Nota: è buona norma, in generale, non modificare le variabili contenenti i dati in ingresso perché può accadere che sia necessario accedere a tali valori in diversi punti del programma.
(Nel caso appena mostrato senza l' ausilio di un' altra variabile S non sarebbe possibile, al termine della computazione, stampare a video il valore del numero inserito -- così come richiesto dalla traccia del problema.)

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main( )
{
      /* dichiarazione delle variabili */
      int i; /* variabile contatore */
      int Q; /* variabile per il numero letto da tastiera */
      int S; /* variabile accumulatore */

      /* ciclo di controllo per l'immissione dati che garantisce Q>0 */
      do
      {
            cout << "\n Inserisci un numero positivo Q: ";
            cin >> Q;          /* legge N dallo standard input */
      } while (Q <= 0);

      S = Q; /* copia del valore del dato in ingresso */
      i = 0; /* inizializzazione del contatore        */
      while (S > 0)
      {
            i = i + 1;
            S = S - i;
      }

      /* all'uscita dal ciclo il valore assunto dalla variabile S */
      /* permette di procedere in base a due alternative          */

      if (S == 0)
      {
      /* il valore del contatore i contiene qui il valore del massimo
         degli addendi che compongono il numero triangolare inserito.*/

            cout << "\n " << Q << " = alla somma dei primi ";
            cout << i << " numeri positivi! \n";
      }
      else
      {
            cout << "\n Il numero " << Q;
            cout << " non e' un numero triangolare! \n";
      }

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

Esercizio 5

Scrivere un programma che legge da stdin una sequenza (di lunghezza a priori illimitata) di numeri interi positivi, terminata da 0, e indica, alla fine della sequenza, qual è la lunghezza della massima sottosequenza di numeri consecutivi in ordine crescente.

Esempi:

13 3 8 4 5 1 17 0
Lung. max = 2

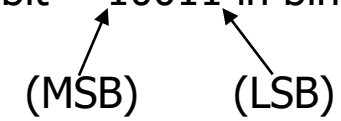
21 19 18 14 9 6 4 3 0
Lung. max = 1

2 1 3 6 8 5 1 12 18 17 0
Lung. max = 4

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib>  // per la funzione system(...)
using namespace std;

int main( ) {
    int valore, precedente = 0, lungAttuale = 0, lungMax = 0;
    cout << "Inserisci una sequenza di int positivi terminata da 0: \n";
    do {

        do {
            cin >> valore;
        } while (valore < 0);

        if (valore != 0 && precedente <= valore)
        {
          lungAttuale += 1;
          if (lungAttuale > lungMax) lungMax = lungAttuale;

        } else {
          lungAttuale = 1;
        }
        precedente  = valore;

   } while (valore != 0);

   cout << endl << "Lunghezza massima delle sottosequenze crescenti =  ";
   cout << lungMax << endl;

   system("PAUSE");
   return EXIT_SUCCESS;
} // end main
```

Esercizio 6

Dato un numero positivo Q, scrivere la sua rappresentazione in binario naturale, indicando anche il minimo numero di bit utilizzato.
Esempio:
Input: 19 in decimale, Output: con 5 bit = 10011 in binario.



1° soluzione

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main(int argc, char *argv[])
{
      int Q, Qaux, current, i, n;
      cout << "\n Numero intero Q non negativo : Q = ";
      cin >> Q;
      Qaux = Q;
      n = 0;
      do {
            Qaux = Qaux / 2;
            n = n + 1;
      } while (Qaux != 0);
      cout << "\nCodifica di Q = " << Q << " con " << n << " bit ";
      for (current = n-1; current >= 0; current--) {
            Qaux = Q;
            for (i=0; i < current; i++) {
               Qaux = Qaux / 2;
            }
            cout << (Qaux % 2);
      }
      cout << endl;

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

2° soluzione

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main(  )
{
      int dec, Q, n, base;
      cout << "\n Numero intero Q non negativo : Q = ";
      cin >> Q;
      dec=0;  // accumulatore
      n=0;    // posizione bit – a fine ciclo conterrà il num. di bit
      base=1; // accumulatore delle potenze di 10 consecutive;10^0=1
      while (Q > 0)
      {
            dec = dec + (Q % 2) * base;
            Q = Q / 2 ;
            n = n + 1 ;
            base = base * 10;
      }
      cout << "\n Numero decimale che emula la sequenza ";
      cout << "binaria di " << n << " bit: " << dec;
      cout << endl;

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

3° soluzione

Idea di soluzione:

se $Q = q_{n-1}2^{n-1} + q_{n-2}2^{n-2} + \ldots + q_1 2 + q_0$, con $q_i = \{0, 1\}$ allora posso confrontare Q con le successive potenze di 2: 1, 2, 2^2, 2^3, … finché risulta soddisfatta la condizione $Q >= 2^j$ con j = 0, 1, 2,….
Quando si verificherà la condizione per cui $Q < 2^n$, l' esponente di tale potenza sarà proprio il numero di bit necessario a rappresentare Q.

Osservando che:

Essendo $Q \geq 2^{n-1}$ (Q rappresentato con *n* bit)
allora $q_{n-1} = 1$ per costruzione

se $(Q - q_{n-1}2^{n-1}) \geq 2^{n-2}$
allora $q_{n-2} = 1$
altrimenti $q_{n-2} = 0$
se $(Q - q_{n-1}2^{n-1} - q_{n-2}2^{n-2}) \geq 2^{n-3}$
allora $q_{n-3} = 1$
altrimenti $q_{n-3} = 0$
...
se $(Q - q_{n-1}2^{n-1} - q_{n-2}2^{n-2} - \ldots - q_0 2^0) \geq 1$
allora $q_0 = 1$
altrimenti $q_0 = 0$

la stesura informale dell' algoritmo

1. inizio Programma
2. leggi il numero intero Q>=0 da convertire
3. inizializza un contatore n (per segnare la posizione dei bit) al valore 0
4. inizializza un accumulatore d, per le potenze di 2, al valore 1
5. esegui
   - 5.1. incrementa di uno il contatore n // …. l'esponente!
   - 5.2. moltiplica per 2 la variabile d
   - 5.3. Finché il numero da convertire è >= d, torna a 5.
6. stampa a video il valore della variabile n // …. qui $d == 2^n > Q$
7. segna la posizione del bit più significativo: n = n-1
8. esegui
   - 8.1. se (Q >= d) allora
     - 8.1.1. stampa a video un carattere "1"
     - 8.1.2. assegna a Q il valore Q - d
   - 8.2. altrimenti stampa a video il carattere "0"
   - 8.3. decrementa di 1 il contatore n
   - 8.4. dividi per 2 la variabile d

Finché n >= 0, torna a 4.
9. fine programma

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main( )
{
      int i,d,n; /* contatore */
      int Q; /* variabile per il numero letto da tastiera */

      /* ciclo di controllo per l'immissione dati che garantisce Q>0 */
      do
      {
            cout << "\n Inserisci un numero positivo Q: ";
            cin >> Q;
      } while (Q <= 0);
      n = 0;
      d = 1;
      do {
            n = n + 1;
            d = d * 2;
      } while (Q > d);

      /* d contiene la piu' piccola potenza di 2 maggiore di Q */
      /* n ha il valore dell'esponente della potenza di 2 */
      /* memorizzata in d e coincide con il numero di bit */
      /* minimo per rappresentare Q. */
      /* n-1 è il */
      /* valore dell'esponente più significativo presente nella */
      /* rappresentazione binaria del numero Q. */
      /* Infatti Q = 2^(n-1) + … */

      cout << "\n" << Q << " in decimale, con " << n << " bit = ";
      n = n - 1;
      d = d / 2;
      do {
            if (Q >= d) {
                  cout << "1";
                  Q = Q - d;
            } else {
                  cout << "0";
            }
            n = n - 1;
            d = d / 2;
      } while ( n >= 0 );

      cout << endl;

      system("PAUSE");
      return EXIT_SUCCESS;
} // end main
```

Esercizio 7

Scrivere un programma che legge un intero positivo `n` da *stdin* e verifica se può essere scomposto nella somma di due quadrati (verifica cioè se $a,b,n \mid a^2+b^2=n$).
Se sì, stampare a video la scomposizione.

Esempi:

2 ==> 2 = 1 + 1 = $1^2 + 1^2$
28 ==> NON SCOMPONIBILE
146 ==> 146 = 25 + 121 = $5^2 + 11^2$

a) Mostrare, quando ve ne è più di una, tutte le diverse scomposizioni dello stesso numero (ad esempio 50 ha due scomposizioni, 1+49 e 25+25, mentre 5525 è il primo numero ad avere ben sei diverse scomposizioni e 8125 è il primo ad averne esattamente cinque).

b) Verificare anche la scomponibilità in somma di tre quadrati.

```
#include <iostream> // inclusione della libreria standard
#include <cstdlib> // per la funzione system(...)

using namespace std;

int main( ) {
  int n,a,b;
  int nonScomponibile = 1;

  cout << "Inserisci un numero: ";
  cin >> n;
  cout << "Scomposizione: \n";

  for (a = 1; a <= n/2; ++a) {      // basterebbe: a < square-root(n/2)
     for (b = 1; b <= n-a*a; ++b) { // basterebbe: b < square-root(n-a*a)
        if (a*a + b*b == n) {
           cout << n << "=" << a*a << "+" << b*b << endl;
           nonScomponibile = 0;
        }
     }
  }
  if (nonScomponibile == 1)
    cout << n << " -> NON SCOMPONIBILE ! " << endl;

  system("PAUSE");
  return EXIT_SUCCESS;
} // end main
```