Libreria di funzioni C# per la gestione degli indirizzi IP v4

Libreria di base

1. Scrivere una libreria di funzioni per gestire gli indirizzi IP v4. In particolare, è richiesto lo sviluppo delle seguenti funzioni:

   a) funzione **controllo**: verifica se una data stringa, ricevuta in ingresso, rappresenta un indirizzo IP in notazione decimale puntata. La funzione genera un'eccezione se la verifica non ha successo.

   Esempio:

   ```
   controllo("192.168.99.2");          // OK
   controllo("176.2");                 // Eccezione
   controllo("3453.0.123.200");        // Eccezione
   ```

   b) funzione **vettore**: riceve una stringa *s* in ingresso contenente un indirizzo IP e converte le sue parti in un vettore di quattro elementi interi. La funzione genera un'eccezione se l'indirizzo contenuto in *s* non è valido.

   Esempio:

   ```
   v = vettore("192.168.99.2");        // v = [192, 168, 98, 2]
   v = vettore("192.168.99.302"); // Eccezione
   ```

   c) funzione **binario**: converte la rappresentazione di un indirizzo IP, ricevuto in ingresso, da notazione decimale puntata a binaria. Restituisce un intero senza segno a 32 bit contenente il risultato della conversione. La funzione genera un'eccezione se l'indirizzo IP non è valido.

   Esempio:

   ```
   b = binario("192.168.99.2");        // b = 3232260866  (0xC0A86302)
   ```

   d) funzione **controllo_mask**: verifica se una stringa, ricevuta in ingresso, rappresenta una netmask in notazione decimale puntata. La funzione genera un'eccezione se la verifica non ha successo.

   Esempio:

   ```
   controllo_mask("255.255.128.0");          // OK
   controllo_mask("255.128.255.0");          // Eccezione
   controllo_mask("255.255.127");       // Eccezione
   ```

   e) funzione **controllo_cidr**: verifica se una stringa, ricevuta in ingresso, rappresenta un indirizzo in notazione CIDR (esempio: 192.168.1.5/24). La funzione genera un'eccezione se la verifica non ha successo.

   Esempio:

   ```
   controllo_cidr("18.89.54.87/8");          // OK
   controllo_cidr("18.89.54.87/33");    // Eccezione
   ```

```
controllo_cidr("18.89.54.87");       // Eccezione
```

f) funzione **vettore_cidr**: riceve una stringa *s* in ingresso contenente un indirizzo IP in notazione CIDR e converte le sue parti in un vettore di cinque elementi interi (l'ultimo elemento indica il suffisso CIDR). La funzione genera un'eccezione se *s* non è una rappresentazione valida.

Esempio:

```
v = vettore_cidr("192.168.99.2/24"); // v = [192, 168, 98, 2, 24]
v = vettore_cidr("19.18.129.2/54");  // Eccezione
```

g) funzione **binario_cidr**: converte un indirizzo IP, ricevuto in ingresso, da notazione CIDR a binaria. Restituisce: un intero senza segno a 32 bit contenente il risultato della conversione del solo indirizzo IP; un intero senza segno a 8 bit contenente il suffisso CIDR (utilizzare un parametro di output). La funzione genera un'eccezione se il dato in ingresso non è valido.

Esempio:

```
b = binario_cidr("192.168.99.2/24", c);      // b = 3232260866  c = 24
```

h) funzione **ndp**: la funzione è definita in tre modi diversi:

i. a un solo parametro intero (*ip*): converte l'intero *ip* nell'indirizzo IP corrispondente in notazione decimale puntata.

Esempio:

```
1. s = ndp(3232260866);       // s = "192.168.99.2"
```

ii. a due parametri interi (*ip*, *mask*): converte l'indirizzo *ip* e la maschera di rete *mask* in una stringa corrispondente all'indirizzo IP in notazione CIDR.

Esempio:

```
1. s = ndp(3232260866, 0xFFFFFF00);       // s = "192.168.99.2/24"
```

iii. a due parametri di tipo stringa (s_*ip*, s_*mask*): converte l'indirizzo s_*ip* e la maschera di rete s_*mask* in una stringa corrispondente all'indirizzo IP in notazione CIDR.

Esempio:

```
s = ndp("192.168.99.2", "255.255.255.0");  // s = "192.168.99.2/24"
```

i) funzione **netmask**: restituisce la netmask (in binario) a partire da un indirizzo IP non notazione CIDR.

Esempio:

```
nm = netmask("192.168.10.121/24");    // nm = 4294967040 (0xFFFFFF00)
```

j) funzione **network**: restituisce l'indirizzo di network (in binario) dati i valori interi dell'indirizzo IP e della netmask di un host.

Esempio:

```
ip = binario("172.16.34.67");
m = netmask("255.255.0.0");
```

```
n = network(ip, m);
str = ndp(n);                  // str = "172.16.0.0"
```

k) funzione **broadcast**: restituisce l'indirizzo di broadcast (in binario) dati i valori interi dell'indirizzo IP e della netmask di un host.

Esempio:

```
ip = binario("172.16.34.67");
m = netmask("255.255.0.0");
bc = broadcast(ip, m);
str = ndp(bc);                 // str = "172.16.255.255"
```

l) funzione **controllo_network**: la funzione è definita in due modi diversi:

i. a un solo parametro di tipo stringa (*ip*): verifica se l'indirizzo in notazione CIDR specificato da *ip* è un indirizzo di rete, generando un'eccezione nel caso in cui la verifica non abbia successo.

Esempio:

```
controllo_network("192.168.23.0/24");  // OK
controllo_network("192.168.23.1/24");  // Eccezione
```

ii. a due parametri di tipo stringa (*ip*, *mask*): verifica se l'indirizzo *ip* avente maschera di rete *mask* è un indirizzo di rete, generando un'eccezione nel caso in cui la verifica non abbia successo.

Esempio:

```
controllo_network("192.168.23.0", "255.255.255.0");  // OK
controllo_network("192.168.23.1", "255.255.255.0");  // Eccezione
```

m) funzione **classe**: restituisce un carattere corrispondente la classe di appartenenza di un indirizzo IP (scrivere la funzione in due modi diversi: in uno l'indirizzo in input è un numero intero, nell'altro è una stringa).

Esempio:

```
cl = classe("192.168.23.0");   // cl = 'C'
cl = classe("253.54.219.123"); // cl = 'E'
```

n) funzione **privato**: restituisce *true* se l'indirizzo IP passato come parametro è privato (scrivere la funzione in due modi diversi: in uno l'indirizzo in input è un numero intero, nell'altro è una stringa).

Applicazioni

2. [**analisi**] – Scrivere un’applicazione console che, dopo aver richiesto l’inserimento da tastiera di un indirizzo ip in notazione decimale puntata, verifichi la validità del dato immesso e riporti le seguenti informazioni: classe di appartenenza; netmask; indirizzo di network e di broacdast; tipo di indirizzo (pubblico o privato).
3. [**stessarete**] – Scrivere un’applicazione console che, dopo aver richiesto l’inserimento da tastiera di due indirizzi ip in notazione decimale puntata, verifichi se tali indirizzi appartengono alla medesima rete.
4. [**piano1**] – Scrivere un’applicazione console che richieda un indirizzo di rete in notazione CIDR e visualizzi un piano di indirizzamento per la rete riportando le seguenti informazioni: indirizzo di rete; netmask; intervallo di indirizzi per i computer; indirizzo di gateway; indirizzo di broadcast.
5. [**piano2**] – Si desidera generare automaticamente un piano di indirizzamento per una rete privata di N computer. Scrivere un’applicazione console che: richieda in input il numero N ($N < 1\,000\,000$); individui la più piccola rete *classful* in grado di fornire gli indirizzi necessari; visualizzi il piano di indirizzamento per i computer; mostri la percentuale di utilizzo.