Zadanie 1

Napisz program, który wczytuje w pętli ciąg liczb całkowitych do momentu, gdy użytkownik poda liczbę 0, która jest tylko sygnałem końca danych i nie jest dalej brana pod uwagę. Następnie program wypisuje wartości najmniejszego i największego elementu wczytanego ciągu oraz liczbę wystąpień tych wartości w całym ciągu.
Na przykład dla ciągu $(7, 3, -24, 2, 7, -24, 7, 2, 0)$ program powinien wypisać

```
Min = -24 2 razy
Max = 7 3 razy
```

Uwaga: nie wolno stosować żadnych tablic ani innych kolekcji!

Zadanie 2

Primorial nieujemnej liczby całkowitej N, oznaczany N#, jest zdefiniowany jako iloczyn wszystkich liczb pierwszych mniejszych lub równych N. Zakłada się przy tym, że z definicji $0\# = 1\# = 1$. Napisz funkcję obliczającą primorial n.

Na przykład, następujący program

download *PrimorialP.cpp*

```cpp
#include <iostream>
#include <iomanip>     // formatowanie wydruku
#include <stdexcept>   // może być przydatne
#include <cstdint>     // uint64

bool isPrime(int n) {  // może być przydatne
    // ...
}
uint64_t primorial(int n) {
    // ...
}

int main() {
    std::cout << " N         primorial(N)\n";
    for (int n = 0; n <= 58; ++n)
        std::cout << std::setw(2)  << n
                  << std::setw(21) << primorial(n) << '\n';
}
```

powinien wydrukować

```
 N         primorial(N)
 0                    1
 1                    1
 2                    2
```

3	6
4	6
5	30
6	30
7	210
8	210
9	210
10	210
11	2310
12	2310
13	30030
14	30030
15	30030
16	30030
17	510510
18	510510
19	9699690
20	9699690
21	9699690
22	9699690
23	223092870
24	223092870
25	223092870
26	223092870
27	223092870
28	223092870
29	6469693230
30	6469693230
31	200560490130
32	200560490130
33	200560490130
34	200560490130
35	200560490130
36	200560490130
37	7420738134810
38	7420738134810
39	7420738134810
40	7420738134810
41	304250263527210
42	304250263527210
43	13082761331670030
44	13082761331670030
45	13082761331670030
46	13082761331670030
47	614889782588491410

```
48   614889782588491410
49   614889782588491410
50   614889782588491410
51   614889782588491410
52   614889782588491410
53 14142414403480493114
54 14142414403480493114
55 14142414403480493114
56 14142414403480493114
57 14142414403480493114
58 14142414403480493114
```

Uwaga: Funkcja primorial bardzo szybko rośnie, dlatego jako typ zwracany wybrany został **uint64_t**. Nawet wtedy największa liczba n, dla której n# mieści się w zmiennej tego typu to 58 (największa wartość **uint64_t** to 18446744073709551615).

Zadanie 3

Napisz i przetestuj następujący zestaw funkcji *rekurencyjnych* (w ich treści nie może być żadnych pętli!)

- `int gcdRec(int a, int b);` — zwraca największy wspólny dzielnik dwóch liczb naturalnych (całkowitych dodatnich);
- `int sumDigits(int n);` — zwraca sumę (dziesiętnych) cyfr podanej liczby naturalnej (e.g., 34→7);
- `int numDigits(int n);` — zwraca liczbę (dziesiętnych) cyfr podanej liczby naturalnej (e.g., 34→2);
- `void printOddEven(int n);` — drukuje w jednej linii
  - dla n parzystego – wszystkie liczby parzyste poczynając od 2 aż do n;
  - dla n nieparzystego – wszystkie liczby nieparzyste od 1 aż do n.
- `void hailstone(int n);` — drukuje w jednej linii wszystkie liczby ciągu Collatza (patrz niżej) od n aż do 1.

Ciąg Collatza, znany też jako ciąg Ulama lub „gradowy” (ang. *hailstone*), to ciąg dla którego pierwszy wyraz a_0 jest dodatnią liczbą całkowitą, a kolejne wyrazy są wyliczane ze wzoru

$$a_{n+1} = \begin{cases} a_n/2 & \text{jeśli } a_n \text{ jest parzyste,} \\ 3a_n + 1 & \text{jeśli } a_n \text{ jest nieparzyste.} \end{cases}$$

Hipoteza Collatza (wciąż nieudowodniona) mówi, że niezależnie od wartości pierwszego elementu a_0, po skończonej liczbie kroków ciąg osiągnie wartość 1 (i potem będzie już cyklicznie przybierał wartości 1→4→2→1...).
Następujacy program, po zdefiniowaniu wszystkich funkcji

download *SimpleRekur.cpp*

```cpp
#include <iostream>

int gcdRec(int a, int b);
int sumDigits(int n);
int numDigits(int n);
void printOddEven(int n);
void hailstone(int n);

int main() {
    std::cout << "gcdRec(12, 42) = " <<
                 gcdRec(12, 42) << std::endl
              << "gcdRec(12, 25) = " <<
                 gcdRec(12, 25) << std::endl;
    std::cout << "sumDigits(123) = " <<
                 sumDigits(123) << std::endl
              << "sumDigits(971) = " <<
                 sumDigits(971) << std::endl;
    std::cout << "numDigits(12345) = " <<
                 numDigits(12345) << std::endl
              << "numDigits(971) = " <<
                 numDigits(971) << std::endl;
    std::cout << "printOddEven(15): ";
    printOddEven(15);
    std::cout << std::endl;
    std::cout << "printOddEven(14): ";
    printOddEven(14);
    std::cout << std::endl;
    std::cout << "hailstone(13): ";
    hailstone(13);
    std::cout << std::endl;
}
```

powinien wypisać

```
gcdRec(12, 42) = 6
gcdRec(12, 25) = 1
sumDigits(123) = 6
sumDigits(971) = 17
numDigits(12345) = 5
numDigits(971) = 3
printOddEven(15): 1 3 5 7 9 11 13 15
printOddEven(14): 2 4 6 8 10 12 14
hailstone(13): 13 40 20 10 5 16 8 4 2 1
```
