Aufgabe
Datentypen, Schleifen

1. Aufgabe *Lucas-Lehmer-Test für Mersenne-Primzahlen*
Mersenne-Primzahlen sind Zahlen der Form

$$M_p = 2^p - 1 \tag{1}$$

für die m_p prim ist. Mersenne-Primzahlen besitzen viele interessante Eigenschaften, u.a. kann man mit dem einfach zu implementierenden Lucas-Lehmer-Test bei relativ geringem Rechenaufwand prüfen, ob sie prim sind. Die Aussage des Tests ist folgende:

> Für ungerade Primzahlen p sei die Folge $S(k)$ definiert durch $S(1) = 4$, $S(k+1) = S(k)^2 - 2 \mod M_p$ mit der Modulo-Funktion $a \mod b$ (Rest der Division $a : b$).
> Dann gilt: M_p ist genau dann eine Primzahl, wenn $S(p–1) = 0$, also $S(p-1)^2–2$ durch M_p teilbar ist.

Wir wollen diesen einfachen Test in einem kurzen C++-Programm umsetzen.

a) Überlegen Sie sich zunächst, welche Datentypen und welche Bibliotheken (`#include <...>`) Sie benötigen.

b) Welche Art von Schleife bietet sich an, um die Folge $S(k)$ zu berechnen?

c) Schreiben Sie ein kurzes Programm, das den Lucas-Lehmer-Test für $p = 7$ durchführt, geben sie u.a. auch M_p aus.

d) Um auch andere Mersenne-Primzahlen zu testen, sollte das Programm eine Eingabe besitzen. Wie kann diese aussehen?

e) Die Funktion `pow` ist eigentlich für Gleitkommazahlen bestimmt. Mit welchem Konstrukt kann man sie ersetzen?

Lösung

a) Man benötigt eigentlich nur Ganzzahl-Datentypen wie `int` oder `unsigned long`. Insbesondere arbeitet der Test auf großen natürlichen Zahlen, somit ist `unsigned long` am besten geeignet für die Darstellung von $S(k)$. Die bisher größte gefundene Mersenne-Primzahl (51.) wurde für $p = 82\,589\,933$ (7.12.2018) im Rahmen von *GIMPS* [1] ermittelt, d.h. für p und damit auch für k sind `int` oder `unsigned int` ausreichend.
Um die benötigte Mersenne-Zahl $M_p = 2^p - 1$ zu berechnen, ist es zweckmäßig, die mathematische Bibliothek `cmath` einzubinden, um so die Potenzfunktion `pow(x,y)` nutzen zu können.
Für die Ausgabe kann die `iostream`-Bibliothek eingebunden werden, damit das Ergebnis mit `cout <<` im Terminal ausgegeben wird.

[1] `https://www.mersenne.org/download/`. Das von GIMPS verwendete Programm Prime95 kann auf Rechnern als "Stresstest" verwendet werden, da es CPU und Speicher maximal fordert.

b) Die Folge läuft von $k = 1$ bis $k = p - 1$, wobei $S(1) = 4$ ist. Man kann also den Fall $k = 1$ schon bei der Deklaration von S als `unsigned long` behandeln:
```
unsigned long s = 4 ;
```
und nutzt dann eine `for`-Schleife der Form
```
for (unsigned int k = 2 ; k < p ; k++) { ... }
```
für $k > 1$.

c) Ein kurzes Programm (ohne Kommentare) könnte wie folgt aussehen:

```
#include <iostream>
#include <cmath>
using namespace std ;

int main() {

  unsigned int p = 7 ;
  unsigned long m ;
  unsigned long s = 4 ;

  m = pow(2, p) - 1 ;
  cout << "p = " << p << " m_p = " << m << endl ;

  for (unsigned int k = 2 ; k < p ; k++) {
      s = ( s*s  - 2) % m ;
      cout << "k = " << k << " s(k) = " << s << endl ;
  }
  if (s == 0) cout << m << " is prim"     << endl;
  else        cout << m << " is not prim" << endl ;

  return 0 ;
}
```

d) Z.B. kann man mit `cin >> p` einen Wert einlesen. Dann sollte man aber auch testen, ob das eingegebene `p` überhaupt eine natürliche Zahl ist. Z.B.

```
int p ; // jetzt unsigned !
cout << "p eingeben: " ;
do     cin >> p ; // Einlesen
while (p <= 0)  ; // Erneutes Einlesen, falls p nicht natuerlich
```

e) Um z.B. andere Ganzzahl-Datentypen (etwa BigInteger) zu verwenden, die `pow` nicht implementieren, kann man wiederum eine einfache Schleife verwenden:

```
m = 2 ; // Berechne 2^p fuer p > 2
for (int power = 2 ; power <= p ; power++) {
  m = m * 2 ;
}
```