Zadanie 1 (obowiazkowe), 2 pkt.

Niech $F_k = 2^{2^k} + 1$. Napisać program, który obliczy liczbę cyfr odwrotności (w zapisie dziesiętnym) w $\mathbb{Z}_{F_n}$ dla kolejnych liczb $F_l = 2^{2^l} + 1$ dla $l = 0, 1, \ldots, n-1$. Liczba `n` powinna być wczytywana ze standardowego wejścia. W rozwiązaniu należy

(a) napisać funkcję `inverse(bigInteger a, bigInteger n)`, która zwróci a^{-1} w $\mathbb{Z}_n$ lub (-1) jeśli a nie jest odwracalne – w jej kodzie należy wykorzystać rozszerzony algorytm Euklidesa;

(b) napisać funkcję `fastPower(bigInteger a, bigInteger n)`, która obliczy w szybki sposób a^n.

Przydatna będzie również metoda `get_str()` z klasy `mpz_class (bigInteger)`. Na koniec: trójkę liczb można reprezentować za pomocą klasy

```
class triple
{
   public:
     bigInteger d, x, y;
};
```

Przykładowe uruchomienie dla $n = 5$ powinno dać wynik

```
Liczba F_5 ma 10 cyfr
Odwrotność F_0 ma 10 cyfr
Odwrotność F_1 ma 10 cyfr
Odwrotność F_2 ma 10 cyfr
Odwrotność F_3 ma 10 cyfr
Odwrotność F_4 ma 10 cyfr
```

Zadanie 2 (obowiazkowe), 1 pkt.

W pliku `pseudorandom.cpp` zaimplementować funkcję `pseudoRandomSeq()` tak, aby generowała kolejne liczby pseudolosowe według podanego algorytmu.

(a) Wygenerować ciąg liczb pseudolosowych o długości 20 dla $x_0 = 1, a = 5, c = 0, m = 12$. Czy tego byśmy się spodziewali od generatora liczb pseudolosowych? Czy zmiana x_0 lub a coś tutaj poprawi?

(b) Wykonajmy punkt poprzedni z wartościami $x_0 = 1, a = 17, c = 0$ oraz $m = 512$. Czy wygenerowany ciąg "wygląda" na losowy? Jaki mankament można w nim dostrzec? Czy zmiana x_0 coś tutaj poprawi?

(c) Weźmy teraz $x_0 = 1, a = 22695477, c = 1, m = 2^{32} = 4294967296$[(a)]. Czy teraz ciąg "wygląda" na losowy?

(d) Możemy jeszcze bardziej "zwiększyć losowość". Od drugiego kroku możemy brać (np.) 15 najmniej znaczących bitów (wykorzystać np. przesunięcie bitowe). [(b)]

Wygenerować taki ciąg o długości 100 (z dowolnie wybranym x_0) a następnie (np.) w LibreOffice wygenerować wykres otrzymanych danych. W tym celu

[(a)] Źródło: podana niżej strona Wikipedii.

[(b)] Źródło: j.w. Jest to sposób bardzo podobny do (podanego za Wikipedią) sposobu generowania liczb pseudolosowych w Borland C/C++.

dane wygenerowane przez program należy zapisać do pliku (np. każdą wartość w oddzielnym wierszu), a następnie wczytać je do LibreOffice Calc i wybrać wykres "punktowy (XY)" (Menu: Wstaw $\rightarrow$ Wykres...)

Dla zainteresowanych: Więcej o takich generatorach liczb pseudolosowych można znaleźć np. na Wikipedii: Wikipedia: Linear Congruential Generator

Zadanie 3, 1 pkt.

To zadanie związane jest z poniższym twierdzeniem Lamégo:

Twierdzenie. Jeśli $a, b \in \mathbb{N}_0$ są takie, że $a > b$ oraz `extendedGCD(a,b)` wykonał n wywołań rekurencyjnych, to $b > F_{n+1}$.

Napisać program, który ze standardowego wejścia wczytuje czas (w sekundach) a następnie wypisuje pierwszą parę liczb Fibonacciego F_{n+1}, F_n dla której wykonanie `extendedGCD`(F_{n+1}, F_n) zajmuje więcej czasu niż podano.

Dla przykładu (na moim komputerze) dla $t = 0.0001$

```
F_161 = 1983924214061919432247806074196061
F_162 = 3210056809456107725247980776292056
```

Pomiar czasu w C++ (dla naszych potrzeb) można znaleźć pod adresem `https://www.geeksforgeeks.org/chrono-in-c/`