Rozważamy ciąg Fibonacciego:

F(0) = 0

F(1) = 1

F(n) = F(n-1) + F(n-2) dla n > 1

System Fibonacciego (https://pl.wikipedia.org/wiki/System_Fibonacciego) to binarny, pozycyjny system liczbowy, w którym poszczególnym pozycjom odpowiadają kolejne liczby Fibonacciego. W zapisie liczby nie używa się pierwszych dwóch liczb z ciągu Fibonacciego. Zaczynającemu się od 1 ciągowi cyfr 0 i 1 (tylko takich się używa) an…a3a2 odpowiada liczba an*F(n) + … + a3*F(3) + a2*F(2).
Wyjątkiem jest liczba 0, którą zapisujemy jako 0. Na przykład liczba zapisana w systemie Fibonacciego jako 1000 oznacza szóstą liczbę w ciągu Fibonacciego, czyli F(5) = 5. Inne przykłady to:

1000101 = F(8) + F(4) + F(2) = 21 + 3 + 1 = 25

10010010 = F(9) + F(6) + F(3) = 34 + 8 + 2 = 44

Cyfrę systemu Fibonacciego nazywamy fibitem. Postać unormowana to taki zapis, w którym nie występują dwie jedynki obok siebie (dwa fibity równe jeden).

Należy zaimplementować klasę Fibo reprezentującą liczby zapisane w systemie Fibonacciego. Niech f1, f2 będą obiektami klasy Fibo. Powinny być dostępne następujące operacje:

Fibo f1 - tworzy liczbę 0

Fibo f1(str) - tworzy liczbę na podstawie napisu str, który jest zapisem tej liczby w systemie Fibonacciego (zapis niekoniecznie musi być w postaci unormowanej)

Fibo(n) - tworzy liczbę równą liczbie całkowitej n

Fibo f1(f2) - konstruktor kopiujący

f1 = f2 - przypisanie

f1 + f2 - dodawanie

f1 & f2 - and fibitowy (na postaci unormowanej)

f1 | f2 - or fibitowy (na postaci unormowanej)

f1 ^ f2 - xor fibitowy (na postaci unormowanej)

f1 << n - przesunięcie fibitowe w lewo o n pozycji

f1 += f2

f1 &= f2

f1 |= f2

f1 ^= f2

f1 <<= n

f1 op f2 - operatory porównujące wartości liczbowe reprezentowane przez f1 i f2,
gdzie op to jeden z operatorów: ==, <, <=, != , >, >=

os << f1 - wypisuje f1 na strumień os bez żadnych białych znaków w postaci
unormowanej

f1.length() - zwraca długość zapisu w postaci unormowanej liczby f1

Ponadto należy zaimplementować globalne funkcje:

Zero() - zwraca obiekt reprezentujący liczbę 0

One() - zwraca obiekt reprezentujący liczbę 1

Funkcje Zero i One powinny zwracać obiekty, których nie można modyfikować,
w szczególności następujące konstrukcje powinny być zgłoszone jako błąd
kompilacji:

Zero() += Fibo("10");

```
One() += Fibo("10");
```

Operatory powinny działać również z udziałem argumentów całkowitoliczbowych, ale nie powinny działać z argumentami będącymi napisami. W szczególności przy deklaracjach

```
Fibo f1, f2;
bool b;
```

nie powinny się kompilować konstrukcje typu

```
Fibo f3(true);
Fibo f4('a');
f1 += "10";
f1 = f2 + "10";
b = "10" < f2;
itp.
```

Natomiast poprawne są następujące konstrukcje

```
f1 += 2;
f1 = f2 + 2;
b = 2 < f2;
itp.
```

Należy zadbać o efektywne działanie konstruktorów i operacji w odniesieniu do

obiektów tymczasowych. Wszystkie operatory, metody i funkcje powinny przyjmować argumenty oraz generować wyniki, których typy są zgodne z ogólnie przyjętymi konwencjami w zakresie używania referencji, wartości typu const i obiektów statycznych.

Przykładowy kod demonstrujący użycie klasy Fibo:

```
#include "fibo.h"

#include <cassert>
#include <iostream>

int main() {
    Fibo f;

    assert(f == Zero());
    assert(Fibo(f) == Zero());
    assert(Zero() < One());
    assert(Fibo("11") == Fibo("100"));
    assert((Fibo("1001") + Fibo("10")) == Fibo("1011"));
    assert((Fibo("1001") & Fibo("1100")) == Zero()); // 1100 == 10000
    assert((Fibo("1100") | Fibo("11")) == Fibo("10100")); // 1100 == 10000, 11 == 100
    assert((Fibo("1001") ^ Fibo("1010")) == Fibo("11"));
    assert((Fibo("101") << 3) == Fibo("101000"));

    f = One();
```

```
    f <<= 3;
    assert(f == Fibo("1000"));

    f = One();
    assert(f + Fibo("1") == Fibo("10"));
    assert(f == One());

    Fibo f1("101");
    Fibo f2 = Fibo("101");
    assert(f1 == f2);

    assert(Fibo("11").length() == 3); // 11 == 100

    std::cout << Fibo("11") << std::endl; // prints 100
}
```

Rozwiązanie będzie kompilowane poleceniem

```
g++ -Wall -Wextra -O2 -std=c++17
```

Rozwiązanie powinno zawierać pliki fibo.h oraz fibo.cc, które należy umieścić w repozytorium w katalogu

```
grupaN/zadanie3/ab123456+cd123456
```