Informatik für Mathematiker und Physiker - AS18

Exercise 3: Boolean expressions & Basic loops

Handout: 2. Okt. 2018 06:00

Due: 8. Okt. 2018 23:59

Task 1.5: two-complement integer representation

Open Task (https://expert.ethz.ch/solve/AwbPtawgCCsttbHK8)

Task

Now we assume an architecture using 4-bit arithmetics using two's complement representation of integers. Convert the following binary numbers to decimal numbers (`0b` is a prefix that indicates binary encoding):

```
1.  0b0001
2.  0b0101
3.  0b0111
4.  0b1000
5.  0b1010
6.  0b1111
```

Task 3: From decimal to binary representation

Open Task (https://expert.ethz.ch/solve/STL6kcBcEo8AHGJiJ)

Task

Write a program that inputs a natural, i.e., `unsigned int`, number `n` and outputs the binary digits of `n` in the *correct* order (i.e., starting with the most significant bit). Do not output the leading zeros.

Input

The decimal digits of a non-negative natural number that can fit into an `unsigned int`, in decreasing significance order, without leading zeroes neither separators (like spaces) between digits.

Example:

```
65533
```

Output

The binary digits of the same non-negative natural number, in decreasing significance order, without leading zeroes neither separators between digits.

Example:

```
1111111111111101
```

Task 2: From Natural Language to C++

Open Task (https://expert.ethz.ch/solve/Saufrm6qpuHSsTaES)

Task

Translate the following natural language expressions to C++ expressions.

Example: a is greater than 3 and smaller than 5. $\Longrightarrow$ **Solution**: `a > 3 && a < 5`

1. a greater than b and the difference between a and b is smaller than 15.
2. a is an even natural number greater than 3.
3. a is at most 5 times greater than b and at least 5 times greater than c.
4. Either a is true or b and c are both false.
5. a is false and b is zero.

Task 4b: Fibonacci overflow check

Open Task (https://expert.ethz.ch/solve/cEeTK7Bu9BveR72cb)

Task

Fibonacci numbers are the integers in the following sequence:
$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$ Each number is the sum of the two previous numbers.

Fibonacci numbers grow fast, thus they can easily exceed the value range of a 32-bit number. Think of a general way how you can check whether the result of an addition would exceed the range (overflow) of a 32-bit number **without actually performing the addition (causing the overflow)**.

Write a program that asks the user for an integer n and then prints the first n *Fibonacci numbers*. Print each number on a new line. Use an `unsigned int` (32-bit) to represent the current Fibonacci number. Using the check described above, if calculating the next Fibonacci number would exceed the range representable by an `unsigned int` (32-bit), exit the loop.

Finally, on a new line print the total number of Fibonacci numbers printed x, and the number of Fibonacci numbers requested n in the format: `x of n`.

Input

A natural number n.

Example:

```
3
```

Output

The n first Fibonacci numbers, one per line, nothing else on each output line, except for numbers that exceeed the range representable by unsigned int. On the next output line, exactly `x of n`, where `x` stands for the number of printed Fibonacci numbers and `n` for the number of requested Fibonacci numbers.

Example:

```
0
1
1
3 of 3
```

Task 1: Expression Evaluation

Open Task (https://expert.ethz.ch/solve/ogsndcBpSSY8t6yXK)

Task

Which of the following expressions evaluate to true, which to false?

1. `3 >= 3`
2. `true || false && false`
3. `(true || false) && false`
4. `3 > (1 < true)`
5. `8 > 4 > 2 > 1`
6. `2 < a < 4` (a is a variable of type int)

Task 4a: Fibonacci primes

Open Task (https://expert.ethz.ch/solve/zGmmf3Rf6isR2vREz)

Task

Fibonacci numbers are the integers in the following sequence:
$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$. Each number is the sum of the two previous numbers.

Fibonacci primes are Fibonacci numbers that are also prime numbers. Write a program that asks the user for an integer m and then computes and prints all Fibonacci primes between 0 and m (including). Print each number on a new line.

Finally, on a new line print the total number of Fibonacci primes found.

Input

A natural number m.

Example:

```
14
```

Output

The Fibonnaci primes between `0` and `m`, inclusive, in increasing order, followed by the number of such primes. There should be exactly one number per output line and nothing else.

Example:

```
2
3
5
13
4
```