Documentation Group Project 1 (Assignment06)

Mario Neuner, Leander Decristoforo

April 28, 2025

Contents

1 Problem01: 7001st prime

1.1 Implementation

For this problem we thought of a simple looping strategy: start at 2 and loop through every number. While doing so check for primes and increment a counter for every prime found. When the 7001st prime is found the while loop terminates and the last prime (therefore the 7001st) is assigned to the result variable and printed out at the end of the file. Other variables used are:

- `validation` – for confirming if a number is prime
- `count` – for counting the number of found primes
- `run` – for running through the natural numbers and checking each for primality

1.2 Output

`The program compiles with "gcc -Wall -Wextra -Werror -Wpedantic -std=c18 prime_7001.c -o prime".` When running `./prime` you get the output:
"The 7001st prime is 70663.000000"

Listing 1: 7001st prime in C

```
1  #include <stdio.h>
2  #define NUM 7001
3  // Authors: Mario Neuner, Leander Decristoforo
4  int main(void) {
5  int count = 0, validation = 0; // 0 means prime, 1 not
6  long double result = 0, run = 2;
7  while (count != NUM) {
8  for (int i = 2; (i * i) <= run; i++) {
9  // Check if prime
10 if (((long long)run % i) == 0) {
11 validation = 1;
12 }
13 }
14 if (validation == 0) {
15 count++;
16 result = run;
17 }
18 run++;
19 validation = 0;
20 }
```

```
21  printf("The␣7001st␣prime␣is␣%Lf\n", result);
22  return 0;
23  }
```

2 Problem02: 5-6-numbers

2.1 Implementation

For this problem we started a while loop that breaks when the desired numbers are found. For this to happen we start with a few variables for keeping track of everything, you find the exact usage at the end of this section. Now we startet with n = 165 and m = 144 since we have to find new values where sm = fn. For that we first compute sm and the start the while loop, in which we compute the current fn value and then again loop through every sm value until the first element of sm being ¿= fn. Then we check whether sm is equal to fn and in that case we found our new values. Otherwise we proceed with incrementing n and starting the loop all over again. This takes a while, but after having found the new elements the code prints the new numbers. The variables used are:

- `n` – for tracking the n values
- `m` – for tracking the m values
- `sm` – for tracking the current sm sequence element
- `fn` – for tracking the current fn sequence element

2.2 Output

`The program compiles with "gcc -Wall -Wextra -Werror -Wpedantic -std=c18 5_6_nums.c -o nums"`. When running `./nums` you get the output:
"The next same number are at m = 27693, n = 31977 with fn = sm = 1533776805.000000 (1533776805.000000)"

Listing 2: 7001st prime in C

```c
#include <stdio.h>
#include <stdlib.h>
// Authors: Mario Neuner, Leander Decristoforo

int main(void) {
    // Start at n 166 and m 144
    long n = 165, m = 144;        // 0 == not found, 1 == found
    long double sm = m * (2 * m -1), fn = 0;
    while (1) {
        // Calculate current sequence values
        fn = (n * (3 * n - 1)) / 2;
        // Go through elements of sm until >= fn
        while (fn > sm) {
            m++;
            sm = m * (2 * m -1);
        }
        // Check if sm equal to fn
        if (sm == fn) {
            break;
        }
        n++;
    }
    printf("The next same number are at m = %ld, n = %ld with fn = sm = %Lf (%Lf)\n", m, n, sm, fn);
    return EXIT_SUCCESS;
}
```

3 Problem03: primeseries-generator

3.1 Implementation

Now this problem to be honest was a bit more challenging. For this problem we implemented a first function called "is_prime" for determining whether a given number is prime like we did in problems before.
The second function called largest_n finds for provided integers a, b and c the maximum value of n such that every natural number from 0 to n the formula computes a prime. We did this with a while loop that computes the formula and tests the result with the "is_prime" function and for the case that the result is less than 1 and thus not prime.
In the main function we just looped through every possible values for a, b and c and first assured that c is prime, for the case n = 0. Then we proceeded by computing the largest value for n with the current a, b and c values via the "largest_n" fuction and afterwards compared it to maximum n value. In this way we can be sure to get the correct maximum n value. The program prints the max n value at the end of the program.
The variables used are:

Main:

- `max` – for tracking the maximum n value
- `current_n` – for holding the current n value
- `a,b,c` – for looping and getting the largest n

larges_n:

- `nmax` – for tracking the maximum n value
- `result` – for storing the result of the formula computations

3.2 Output

`The program compiles with "gcc -Wall -Wextra -Werror -Wpedantic -std=c18 primeseriesg.c -o primeseries".` When running `./primeseries` you get the output:
"The maximum value for n is 28"

Listing 3: 7001st prime in C

```c
#include <stdio.h>
#include <stdlib.h>
#define LIMIT 333

// Checks prime, 0 if, 1 if not
int is_prime(int n) {
    if (n < 2) {
        return 0;
    }
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}

// Check for largest n with given a,b,c
long long largest_n(int a, int b, int c) {
    long long nmax = 0;
    long long result = 0;
    while (1) {
        result = (nmax * nmax * nmax) + a * (nmax *
           nmax) + b * nmax + c;
        if (!is_prime(result) || result < 2) {
            return nmax - 1;
        }
        nmax++;
    }
}

int main(void) {
    long long max = 0, current_n = 0;

    // Loop through every possible a,b and c
    for (int a = -LIMIT + 1; a < LIMIT; a++) {
        for (int b = -LIMIT; b < LIMIT; b++) {
            for (int c = -LIMIT; c < LIMIT; c++) {
                if(!is_prime(c)) {
                    continue;
                }
                // Compute largest n for a,b,c
```

```
                    current_n = largest_n(a, b, c);
                    if (current_n > max) {
                        max = current_n;
                    }
                }
            }
        }
        fprintf(stdout, "The maximum value for n is %lld
            \n", max);
        return EXIT_SUCCESS;
    }
```