**CS2263 Assignment 1**

Kohdy Nicholson

fibprime.c:

/\*

    fibprime.c

    Description:

    Program that determines if a number is fib and prime.

    Author:

    Kohdy Nicholson

    Date:

    2020-05-06

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "helpers.h"

int main(int argc, char\*\* argv){

    int lower, upper, prime, fib, result, range\_size;

    double execution\_time = 0.0;

    int input;

    printf("\nPlease enter a lower range value: ");

    input = scanf("%d", &lower);

    if(input != 1){

        printf("Unable to read the value.\n");

        return 1;

    }

    printf("Now, enter an upper range value: ");

    input = scanf("%d", &upper);

    if(input != 1){

        printf("Unable to read the value.\n");

        return 1;

    }

    clock\_t start = clock();

    if(lower > upper){

        printf("Please ensure the lower value is either equal to or lower than the upper value. Exiting...\n\n");

        return EXIT\_FAILURE;

    }

    // Declare array with the proper size.

    range\_size = (upper - lower) + 1;

    for(int i = 0; i < range\_size; i++){

        fib = isfib(lower);

        if(fib == 1){

            prime = isprime(lower);

            if(prime == 1){

                printf("\n%d is a fib prime number.", lower);

            }

        }

        lower++;

    }

    clock\_t finish = clock();

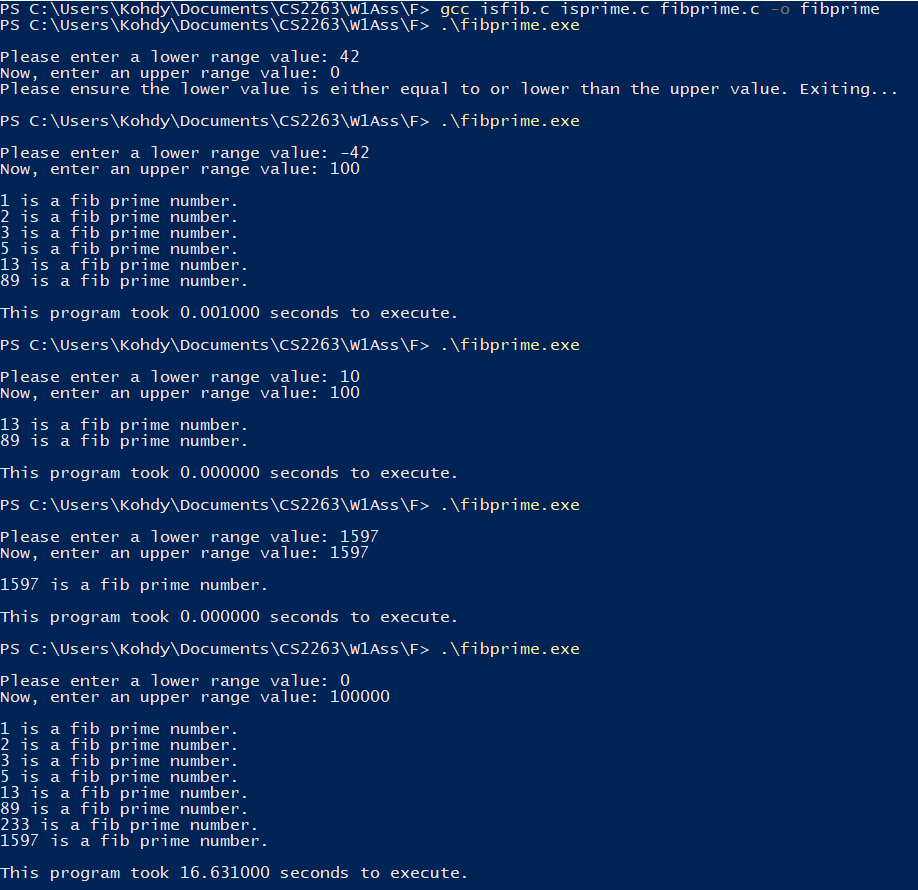
    execution\_time += (double)(finish - start) / CLOCKS\_PER\_SEC;

    printf("\n\nThis program took %f seconds to execute.\n\n", execution\_time);

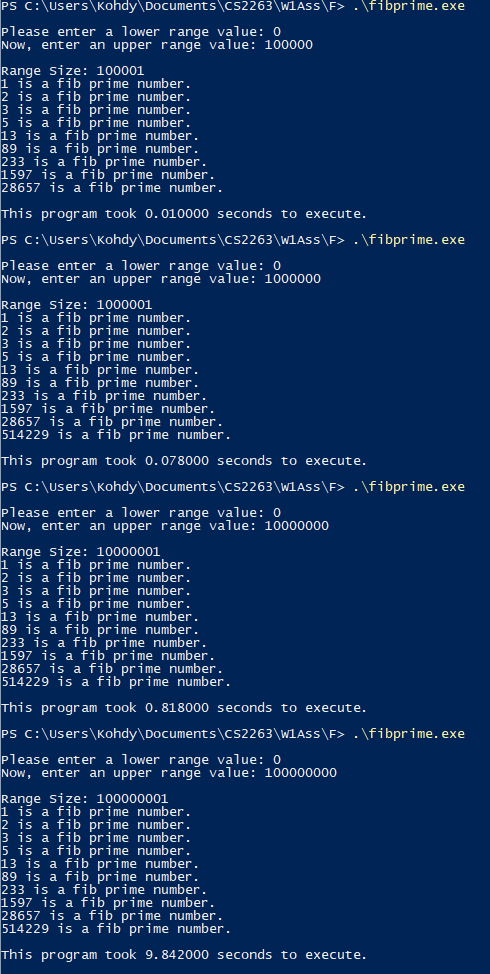
    return EXIT\_SUCCESS;

}

fibprime.exe output:



Faster runtimes:



fibprime\_optimal.c:

/\*

    fibprime\_optimal.c

    Description:

    Program that determines if a number is fib and prime in an optimal way.

    Author:

    Kohdy Nicholson

    Date:

    2020-05-06

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "helpers.h"

int main(int argc, char\*\* argv){

    int lower, upper, input, prime;

    int fib\_num, next\_fib\_num, i, j;

    int arr[255];

    double execution\_time = 0.0;

    printf("\nPlease enter a lower range value: ");

    input = scanf("%d", &lower);

    if(input != 1){

        printf("Unable to read the value.\n");

        return 1;

    }

    printf("Now, enter an upper range value: ");

    input = scanf("%d", &upper);

    if(input != 1){

        printf("Unable to read the value.\n");

        return 1;

    }

    clock\_t start = clock();

    if(lower > upper){

        printf("Please ensure the lower value is either equal to or lower than the upper value. Exiting...\n\n");

        return EXIT\_FAILURE;

    }

    // Initialize array with fib sequence until we surpass the upper value

    fib\_num = 0;

    next\_fib\_num = 1;

    i = 0;

    while(fib\_num <= upper){

        arr[i] = fib\_num;

        fib\_num = next\_fib\_num;

        next\_fib\_num += arr[i];

        i++;

    }

    j = 0;

    while(j < i){

        if(arr[j] >= lower){

            prime = isprime(arr[j]);

            if(prime == 1){

                printf("\n%d is a fib prime number.", arr[j]);

            }

        }

        j++;

    }

    clock\_t finish = clock();

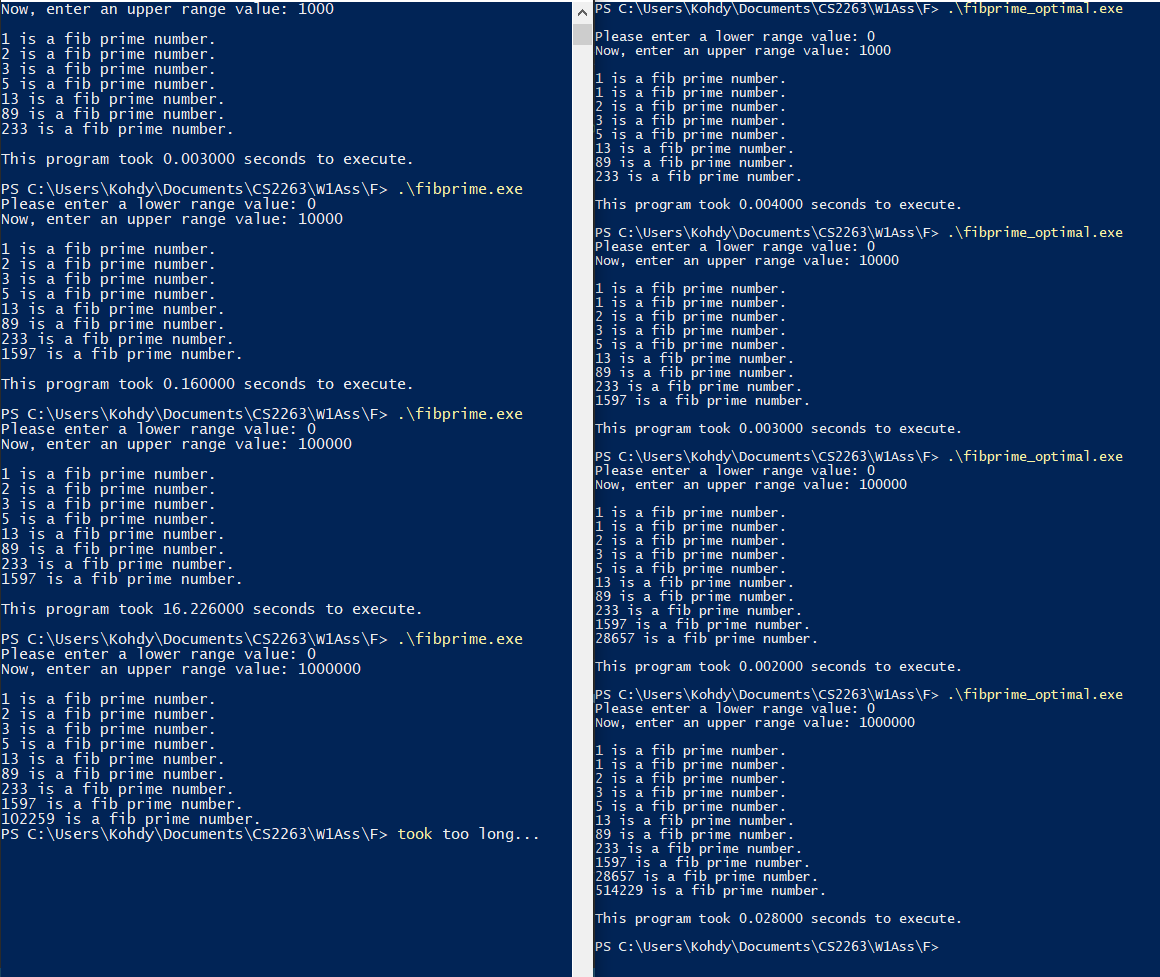
    execution\_time += (double)(finish - start) / CLOCKS\_PER\_SEC;

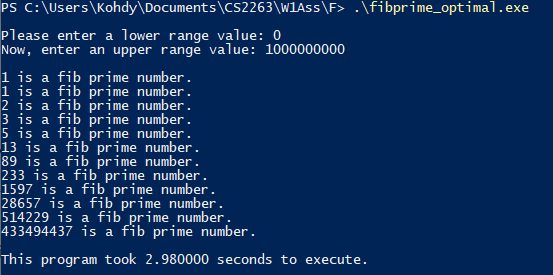
    printf("\n\nThis program took %f seconds to execute.\n\n", execution\_time);

    return EXIT\_SUCCESS;

}

fibprime.exe vs. fibprime\_optimal.exe:





You might notice two things:

1. fibprime\_optimal is MUCH faster. On a range of one million we were able to process all the fib prime numbers in less than one second.
2. fibprime fails to work properly after the number becomes too big. This is because I am using binet’s formula, which takes the square of a number, multiplies it by 5, subtracts and adds 4 and determines if the result is a perfect square. Well, when our number becomes too large, it seems like it cause an integer overflow. Therefore, it does not pick up 28657, or 514229 (also because it was just taking too long).

One solution for this is to simply have a different algorithm for finding if a number is a fib number by creating the fibonacci sequence for everyone number in the range. This would result in longer wait times. I think it is fine the way it is, and demonstrates the idea well enough.

Directory Listing:

