Sushil Khanal

CSCI 301

Section 1

**USER DOCUMENTATION**

Introduction

Recursion is a programming technique in which a function calls itself. It is another way of implementing repetitive computations, which we have previously associated with loops. When called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem.

In this project I have used recursive function in three different circumstance. The first, recursive function that returns the sum of the digits in its non-negative integer argument. The second, recursive function that returns the sum of the integers from low to high. And the final, recursive function that writes the prime factors of an integer to the terminal in ascending order.

Functions

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is main(), and all the most trivial programs can define additional functions.

digisum() returns the sum of the digits in its non-negative integer argument.

Range() returns the sum of the integers from low to high

Pfactor() writes the prime factors of an integer to the terminal in ascending order.

Data Structure

There is only one data structure used in this project, int.

Example: int pfactor(int f)

1. **ADD UP THE DIGITS OF AN INTEGER**

#include <iostream>

using namespace std;

int digsum(int n)

{

if (n < 10)

return n;

return digsum(n / 10) + n % 10;

}

int main()

{

int n;

cout << "Enter an integer: " << endl;

cin >> n;

cout << "The sum of " << n << " is " << digsum(n) << ".";

return 0;

}

1. **SUM A RANGE OF INTEGERS**

#include <iostream>

#include <cstdlib>

using namespace std;

int range(int low, int high);

int main()

{

int x, y;

cout << "Enter two integers, the first one being small than the second one.";

cin >> x;

cin >> y;

int sumrange = range(x, y);

cout << "The sum of the integers from " << x << " to " << y << " is: " << sumrange;

cout << endl;

}

int range(int low, int high)

{

if (low == high) {

return low;

}

else {

return low + range(low + 1, high);

}

}

1. **PRIME FACTORIZATION**

#include <iostream>

#include <cstdlib>

using namespace std;

int pfactor(int f);

int main()

{

int pnumber;

cout << "Enter a positive integer: ";

cin >> pnumber;

cout << "The prime factors of " << pnumber << " are ";

pfactor(pnumber);

cout << endl;

}

int pfactor(int f)

{

if (f == 1) {

return 0;

}

else {

int i = 2;

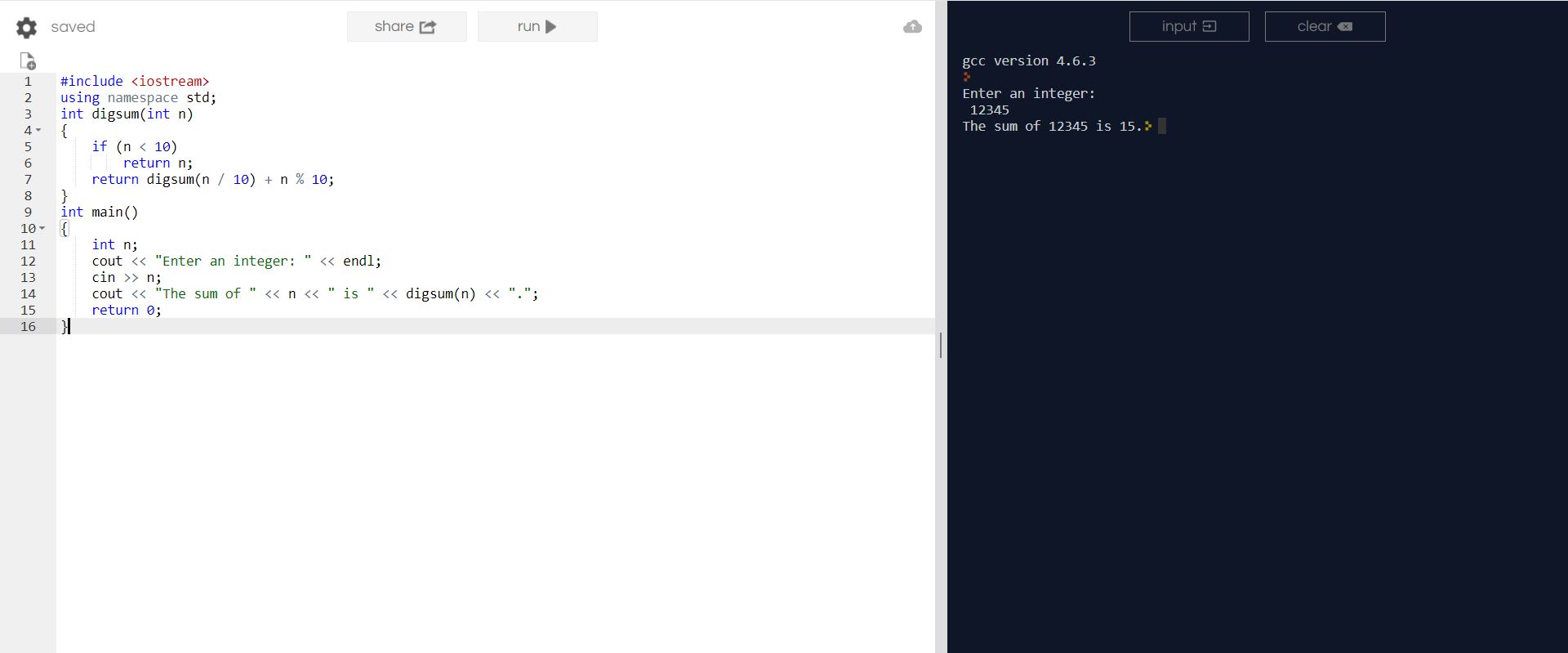
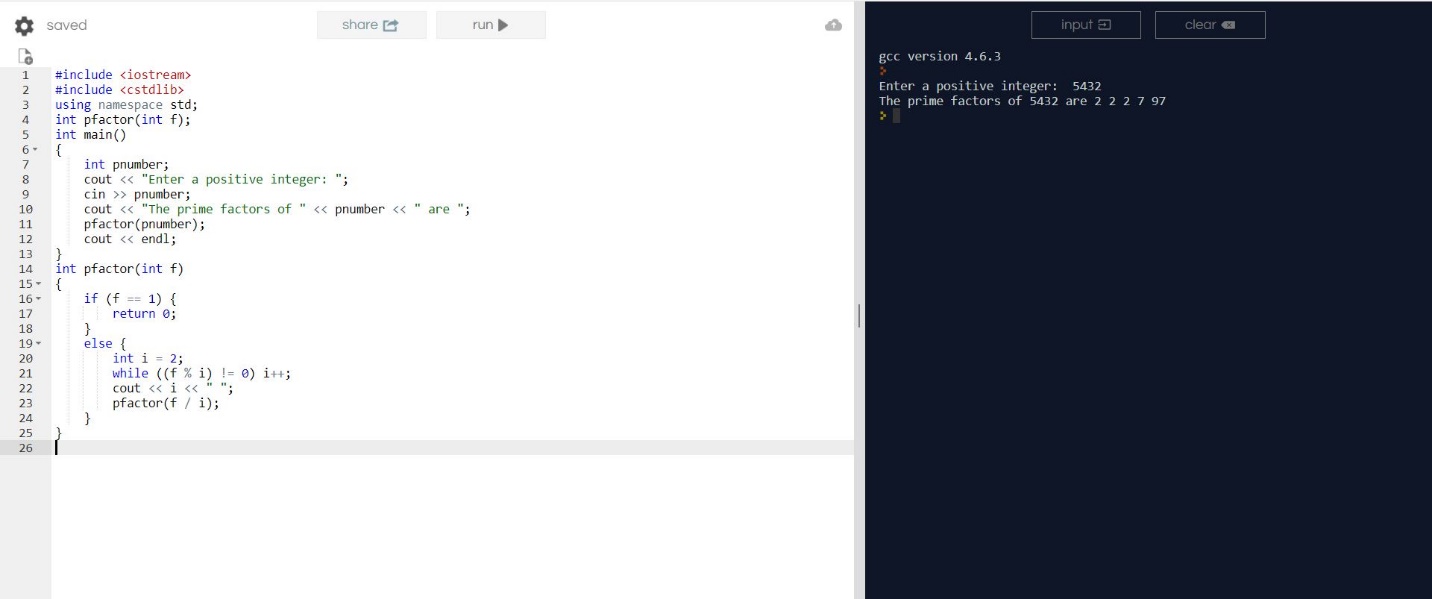
while ((f % i) != 0)

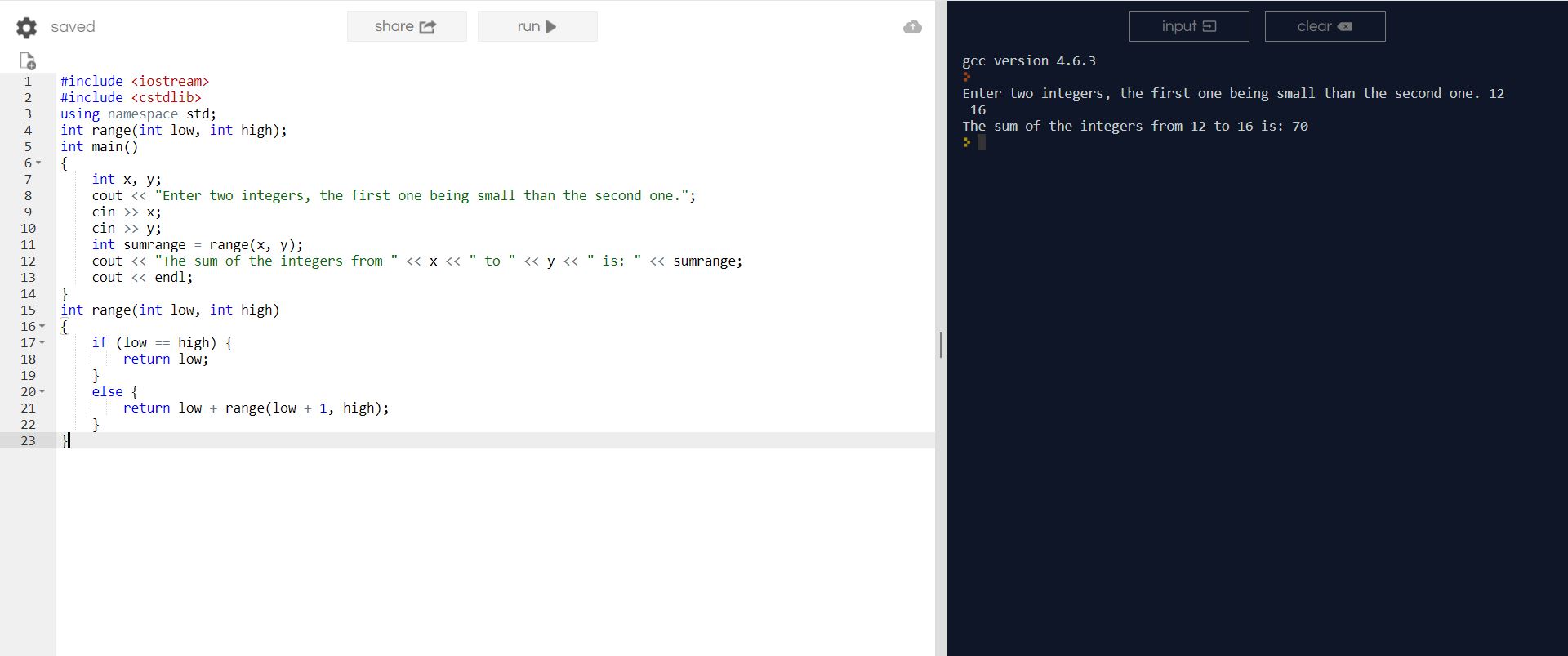
i++;

cout << i << " ";

pfactor(f / i);

}

**TESTING**

****

If we were to modify the function to print the prime factors in descending order, we would no longer get positive output. And the condition “The smallest nontrivial factor of a positive integer is necessarily prime” would not be met.

Summary

This project was quite tricky for me, I couldn’t get my code to work at the beginning. I had to get help from websites like stackoverflow, cplusplus, etc. But at the end, I feel like I now understand recursion and functions better.