Samuel Laron

Project 2

9/13/18

The objective of this project is to demonstrate recursive functions. The program should be able to return the reverse of an integer entered by the user, check an array of integers brought in from a file called vals.dat for a number entered by the user, and the last thing it should be able to do is find the prime factorization of a number entered by the user.

The program will need to consist of 4 functions.

-A recursive function that will reverse the numbers by either printing the number out if it is under 10 or printing the mod 10 of the number then calling the function again with the number divided by 10 in the call. The smaller problem was printing the mod 10 of the number and the base case was when the number got to be under 10.

- A function that will read in numbers from a file and put them into an array

- A recursive function that will take the array and search for a number entered by the user in the max position of the array. If it finds the number in the array it will set a Boolean value to true and display what position it was found in, then if it isn’t in the last position it will call the function again with the position in the array one less. The function should stop searching for the number once it gets to the last position. The smaller problem was searching a position of the array and the base case was once the function got past the final location in the array.

-The final function should first check if the number entered by the user is 1. If it is 1 it should be printed otherwise the function will find the smallest integer that can be divided out of the number passed to the function, then display it. After that it should pass the number entered by the user divided by the integer factored out of it into the function again. It will do this until the number cannot be divided any further. The smaller problem was to find the smallest prime number that could be divided out of the number passed to the function. The base case was once the number passed to the function was equal to 1

//Samuel Larson

//program 2

//9/13/18

#include <iostream>

#include <fstream>

using namespace std;

//this constant integer holds the size of the array

const int size = 8;

//this function prints a number entered by the user backwards

void backwards(int num);

//this function reads numbers from a file into an array

void readin(int arr[size], ifstream&);

//this function checks the array for a number entered by the user and returns if it was found or not

bool check(int num, int array[size], int pos, bool found);

//this function finds the prime factorization of a number entered by the user and prints it

void primefact(int num);

int main()

{

int array[size];

int num;

bool found = false; //a boolean value to hold whether the program found the number entered by the user in the array

ifstream infile;

infile.open("vals.dat"); //the program opens the data file

cout << "Please enter a number" << endl; //the program asks the user to enter a number into the terminal

cin >> num;

backwards(num); //the program calls the function backwards

readin(array, infile);

cout << endl << "Please enter a number" << endl; //the program asks the user to enter a number to search for

cin >> num;

found = check(num, array, size, found); //the program calls the function check

if(found == false){ //if the function returns false it displays that the number wasn't found

cout << num << " was not found in the array" << endl;

}

cout << "Please enter a positive integer" << endl; //the program asks the user to enter a number

cin >> num;

cout << "The prime factorization is" << endl;

if(num == 1)

cout << num;//if the number is 1 it is printed

else

primefact(num); //the function primefact is called

return 0;

}

void backwards(int num){

if(num < 10)

cout << num; //if the number is 1 digit long it prints it and the function returns

else{

cout << num%10; //the last digit in the number is printed

backwards(num/10); //the last digit is taken off the number and the function gets called

}

return;

}

void readin(int array[size], ifstream& infile){

for(int count = 0; count < size; count++){

infile >> array[count]; //the function reads the numbers into the array

}

return;

}

bool check( int num, int array[size], int pos, bool found){

if(array[pos] == num){

found = true;//the number is marked as found in the array and the position is printed out

cout << num << " was found in position " << pos << " of the array" << endl;

}

else if(pos > 0){//if the number hasn't been found it calls the function again to check the next position

found = check(num, array, pos - 1, found);

}

return found;

}

void primefact(int num){

if(num == 1)//if the number is 1 function returns

return;

int factor = 2;

while(num%factor != 0){ //the function finds the lowest number that can be divided out of the number

factor++;

}

cout << factor << " ";

primefact(num/factor); //the function is called again with the number divided by the factor found

return;

}

Samuel Laron

Project 2

9/13/18

To use this program, you must have recursive.cpp in the same directory as a file named vals.dat that contains 8 numbers to be entered into an array.

vals.dat should look something like this

14

66

34

51

44

16

29

33

Once you execute the program you will be prompted to enter an integer. The program will then display the integer backwards. You will be prompted again to enter another integer to search for in the array of numbers used in vals.dat. The final part of the program will ask you to enter another integer and it will calculate the prime factorization of it.

Samuel Laron

Project 2

9/13/18

vals.dat

14

66

34

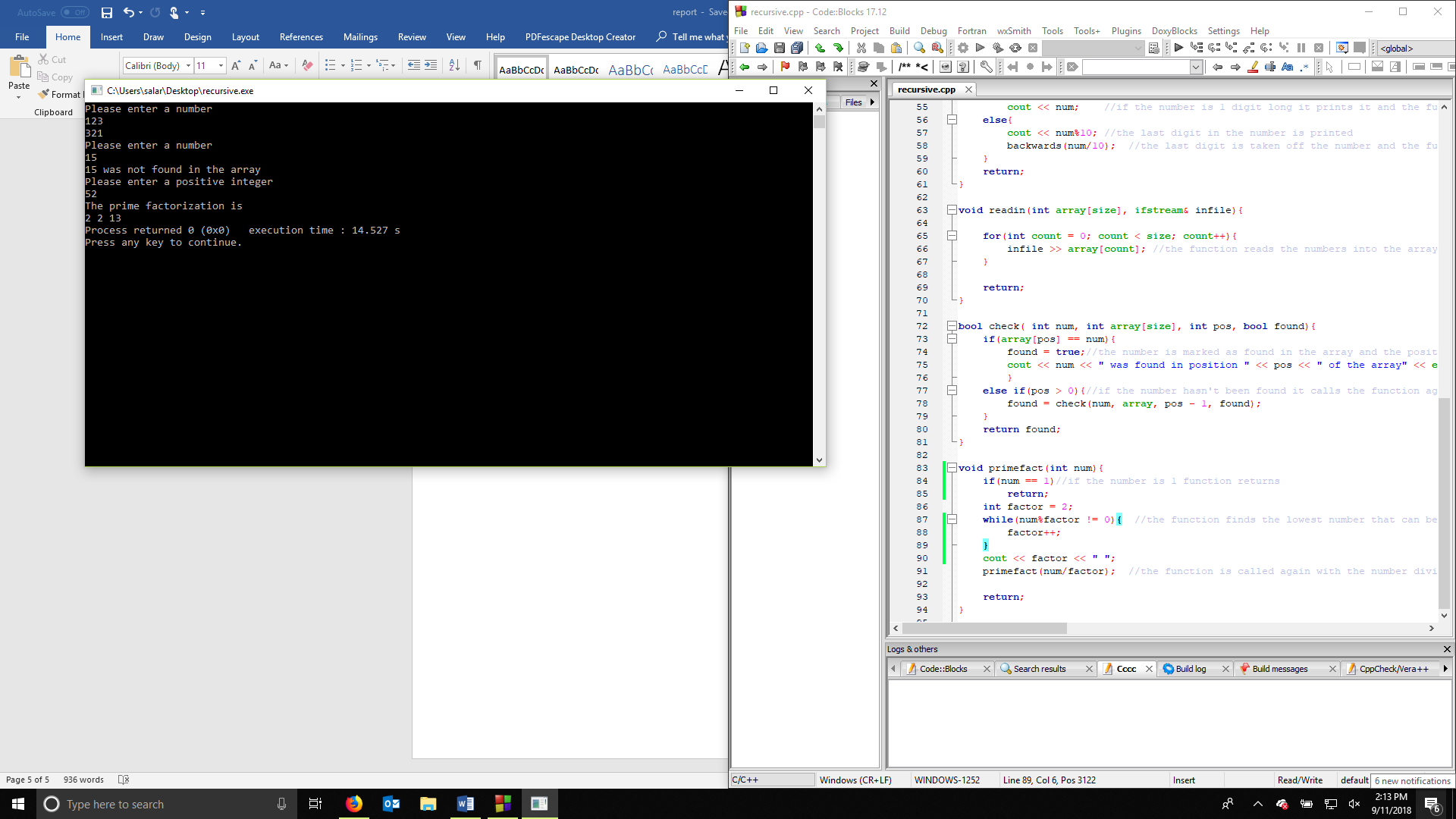
51

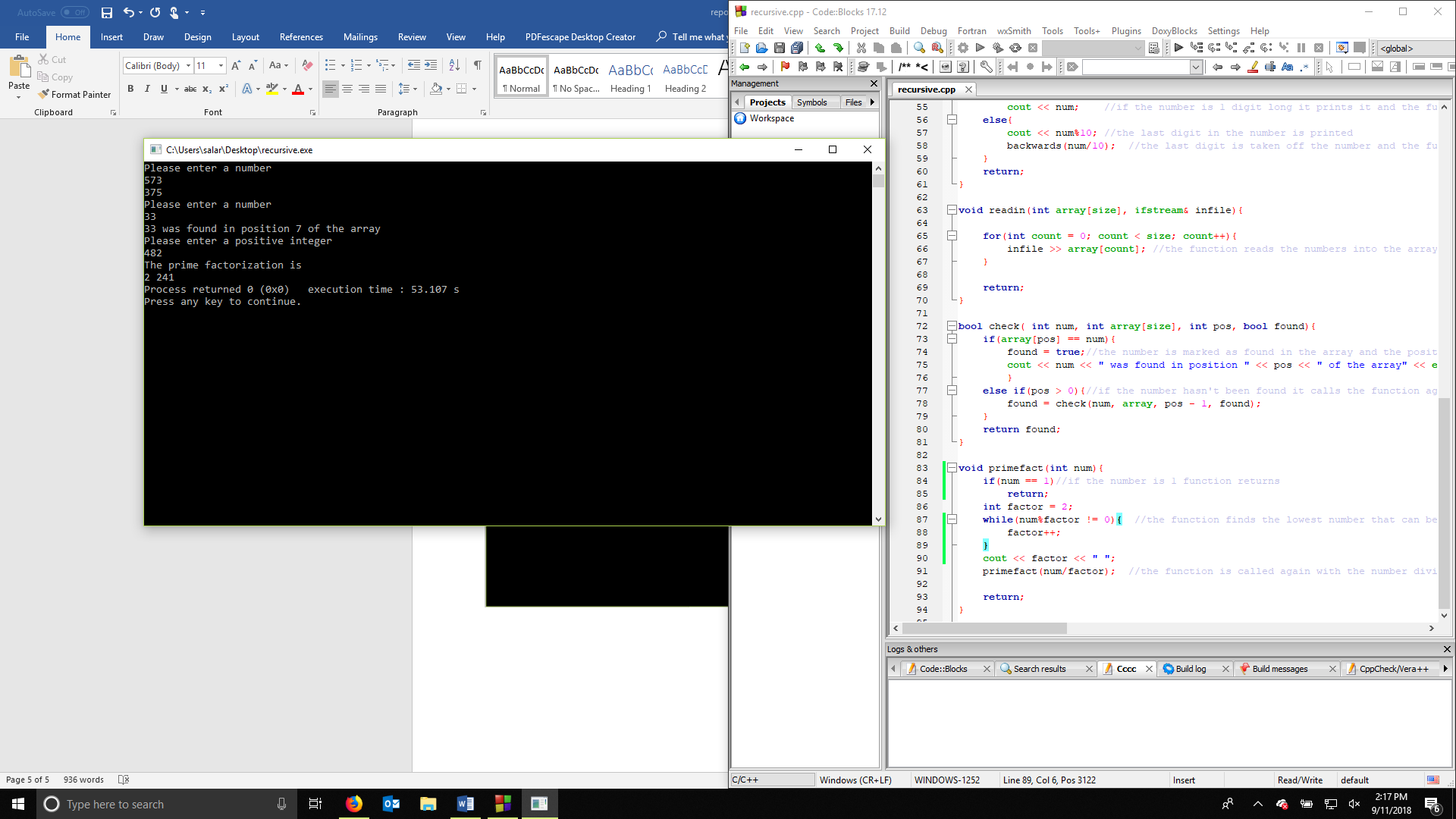
44

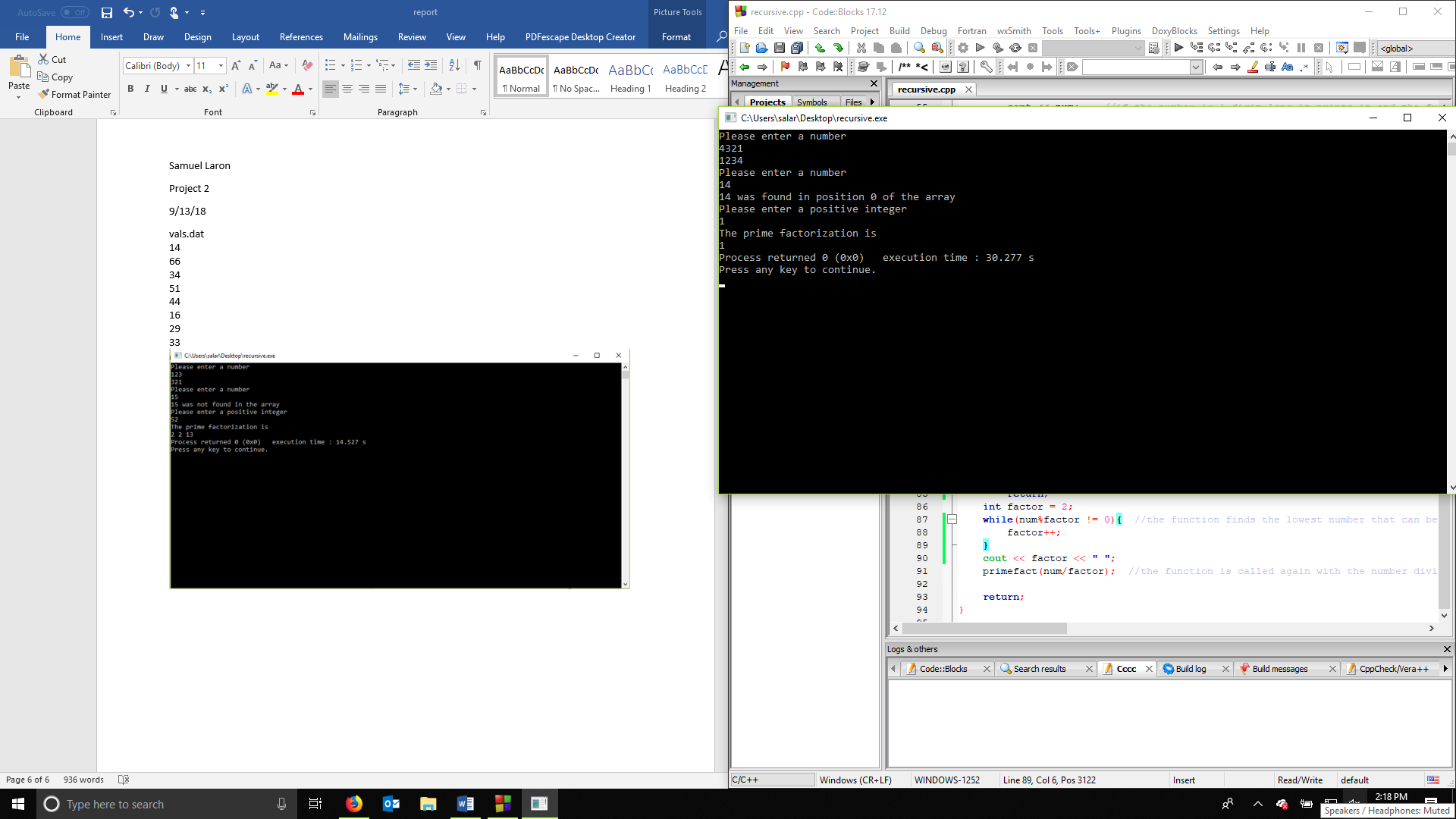
16

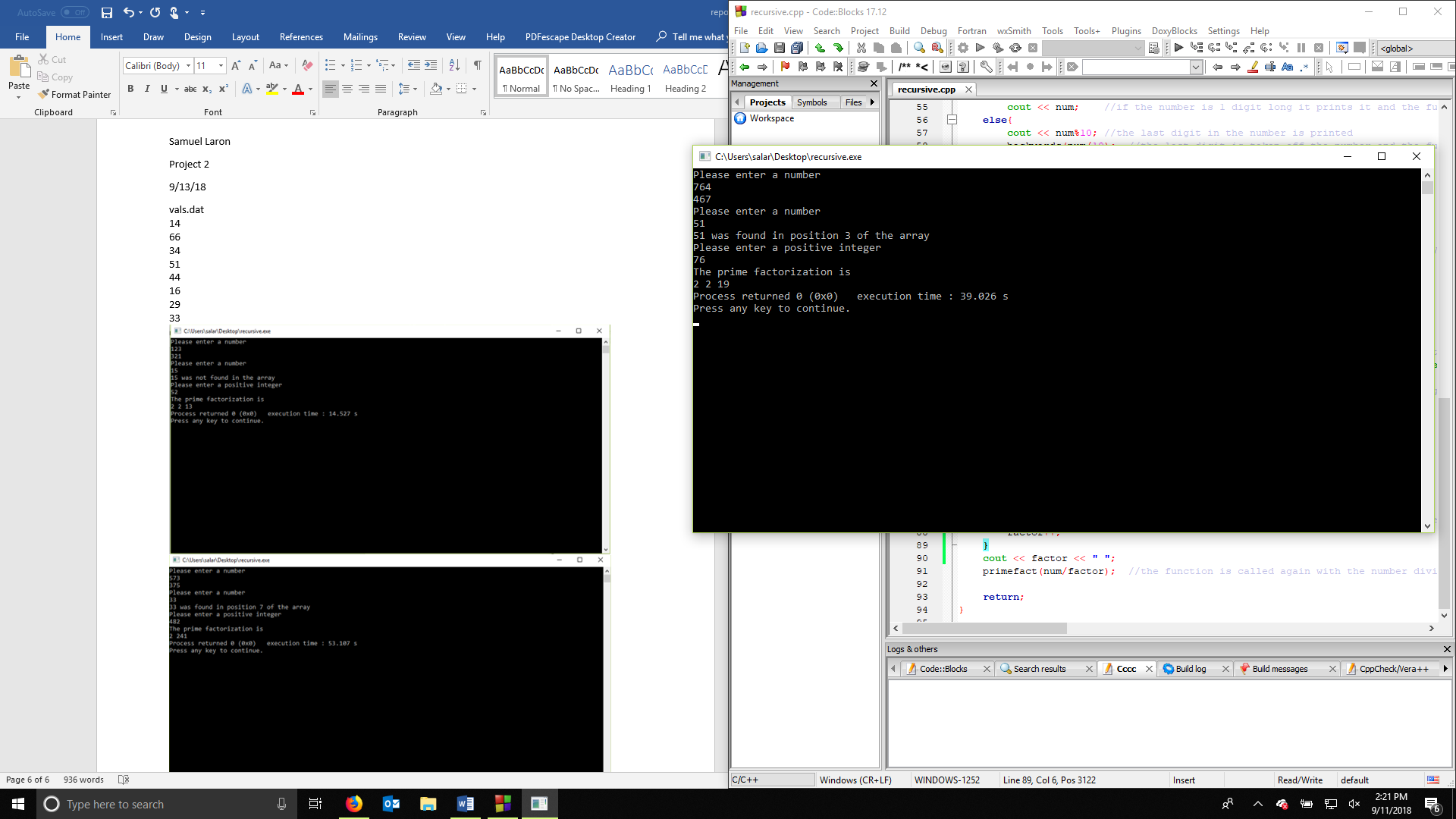
29

33









Samuel Laron

Project 2

9/13/18

With this project I demonstrated how to use recursion to solve 3 different problems. I demonstrated how to make a bigger problem into a smaller one that can be repeated within itself and be terminated with a base case.

The first problem was to display a number in reverse order. My function could be improved by allowing the user to enter a double value instead of just being able to do an integer. The second problem was to search an array of numbers found in a data file for a number entered by the user using recursion. I could have improved upon what I did by using a binary search instead of a linear style search to decrease the time it takes to find the numbers. But that would require the numbers to be sorted before. The final problem I had to solve was to find the prime factorization of a number entered by the user. It could have been done by just using loops, but I feel that recursion was a more efficient way of solving the problem.

To print the prime factors in descending order I would have to switch the print statement and the function call within the function so that it calls the function again before it prints.