Samuel Larson

Program 5

10/11/18

The objective of this program was to rewrite the functions from program 4 recursively to perform the same actions of reading in words from a file and putting them into an ordered list while keeping track of how many of each word there is. The program then it must print out the list and give the user the how many unique words there are.

I will need to rewrite the input(), write(), empty() and length() functions. The smaller problem in each of these is bringing a pointer down the list.

In the insert function it should stop when it finds the word to increment or the next pointer is null to create a new node or increment an existing one accordingly.

The write function should go down the list and print the values in each node until it reaches the end.

The empty function should get to the end of the list and delete nodes going backwards.

The length should get to the end of the list and return plus 1 for each node that exists to find how many unique words there are.

The list destructor should call the empty function because they have the same function.

Each of the functions in this program will need a friend function in the public part of the class to allow the program to call them.

//Samuel Larson

//program 3

//10/4/2018

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

const int MAX = 8;

int length();

class list

{

public:

typedef char item[MAX+1];

typedef int num;

//constructors

list(){first = NULL;} //creates the start of the list

list(const list&);

//destructor - gives back a list's nodes

~list();

//modification member functions

void insert(item entry); //calls the private insert function

//constant member functions

void print(); //calls the private print function

int length(); //calls the private length function

private:

//data members

struct node //structure of a node

{

item word;

num amount;

node\*next;

};

node\*first;

//private function

void insert(item entry, node\*&); //inserts a number to the list

void make\_empty(node\*); //empties the list

int length(node\*); //prints the length

void print(node\*); //prints the list

node\*get\_node(item entry, node\*link) //creates a node

{

node\*temp;

temp = new node;

strcpy(temp->word,entry);

temp->amount = 1;

temp->next = link;

return temp;

};

};

int main()

{

list wordlist;

string filename;

char word[MAX+1];

ifstream infile;

cout << "Please enter a file name" << endl;

cin >> filename;

infile.open(filename); //opens the file m1.dat to take in information

while(infile.good() == false)

{

cout << "file does not exist" << endl

<< "Please enter a file name" << endl;

cin >> filename;

infile.open(filename); //opens the file m1.dat to take in information

}

while(!infile.eof())//inserts more words while it isn't the end of the file

{

infile >> word;

wordlist.insert(word);

}

cout << "Word & Count" << endl;//start of the display of all words

wordlist.print();

cout << "There is " << wordlist.length() << " unique word(s)" << endl;

wordlist.~list();

return 0;

}

//post:list is deconstructed

list::~list()

{

make\_empty(first);

}

//post: list is emptied

void list::make\_empty(node\*p)

{

if(p != NULL)

{

make\_empty(p->next);

delete p;

}

}

void list::insert(item entry)

{

insert(entry, first);

}

//pre:user enters a number to be added post: number is added to the list

void list::insert(item entry, node\*&p)

{

if(p == NULL || strcmpi(entry,p->word) > 0){

p = get\_node(entry, p);

return;

}

if(strcmpi(entry,p->word) == 0)//checks if the words are the same

p->amount = p->amount + 1;//increments the counter of that word

else

insert(entry, p->next);

}

//post:private function print is called

void list::print()

{

print(first);

}

//post:function prints the list

void list::print(node\*p)

{

if(p != NULL){

cout << p->word << " " << p->amount << endl;

print(p->next);

}

}

//post:returns the length of the list

int list::length()

{

return length(first);

}

//post:function returns the length of the list

int list::length(node\*p)

{

if(p == NULL)

return 0;

else

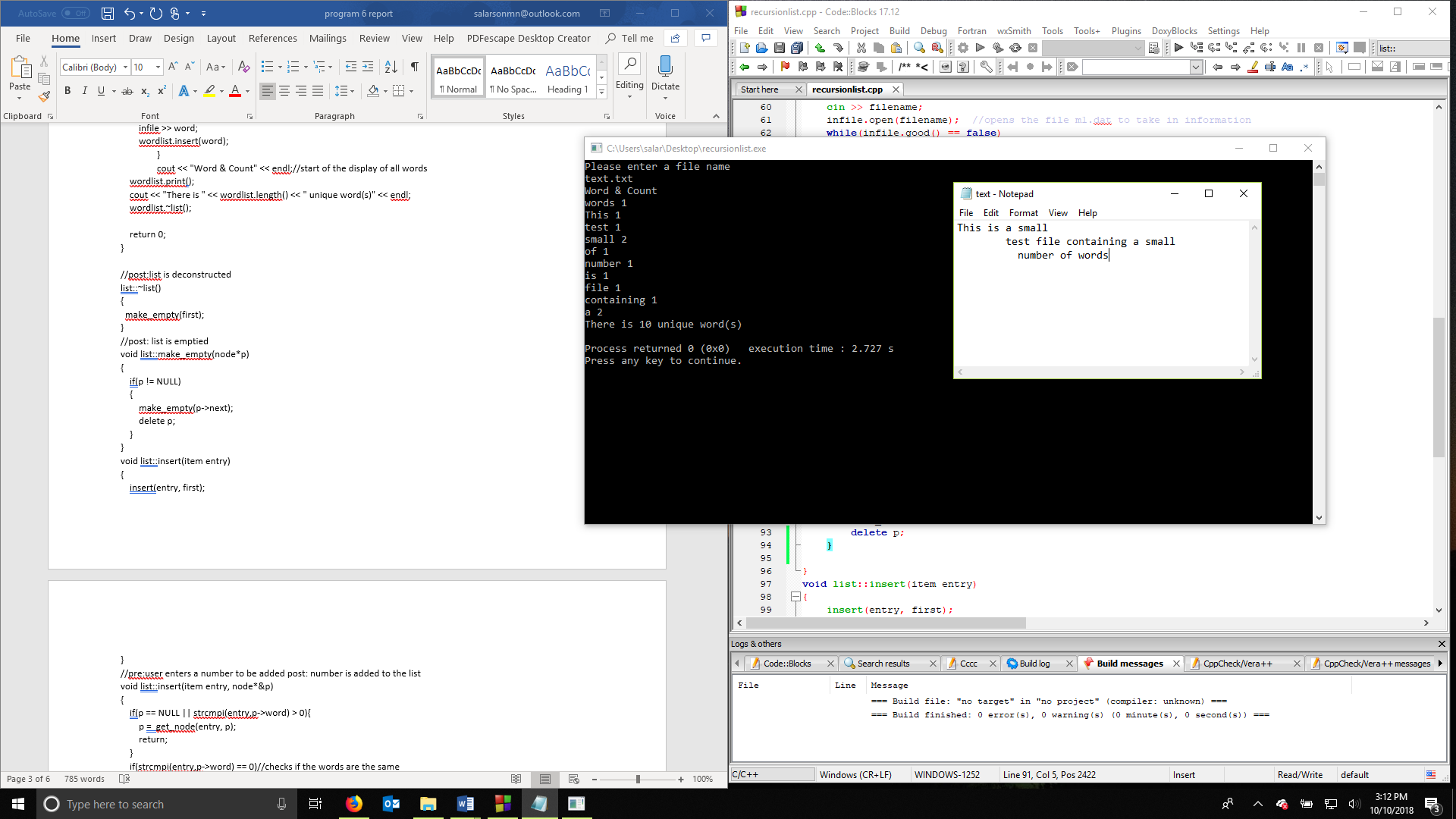
return length(p->next)+1;

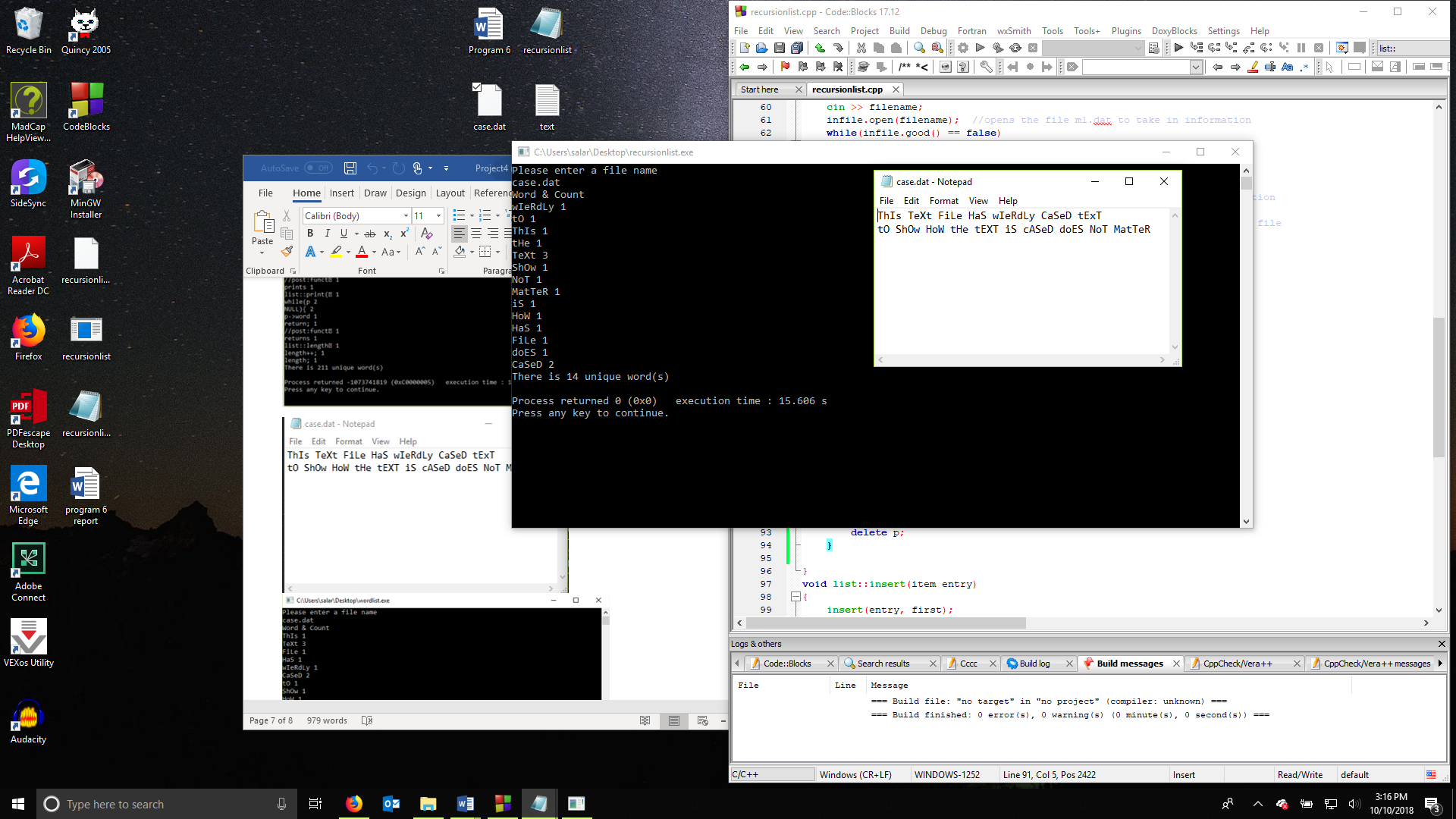
}

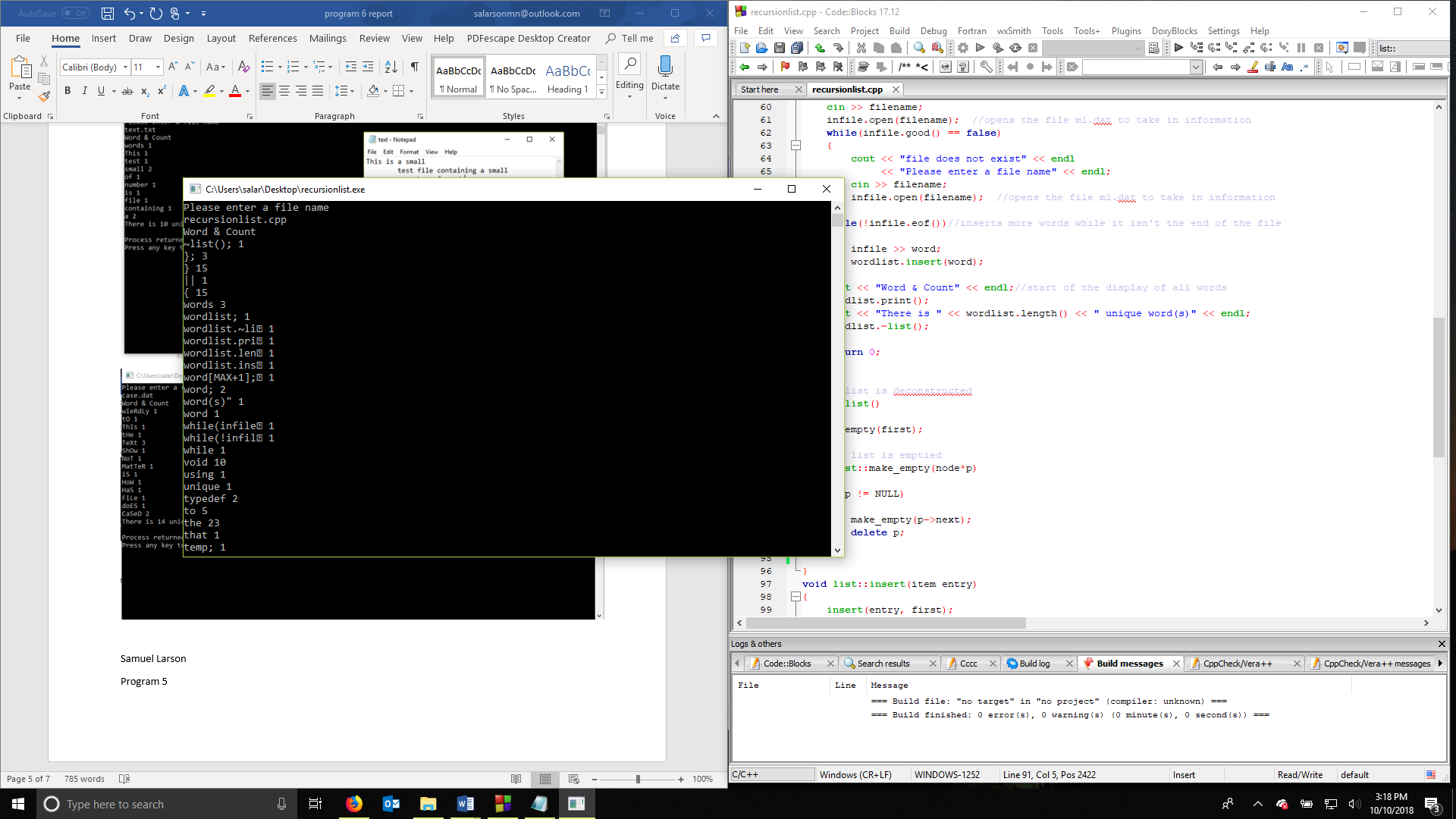
Samuel Larson

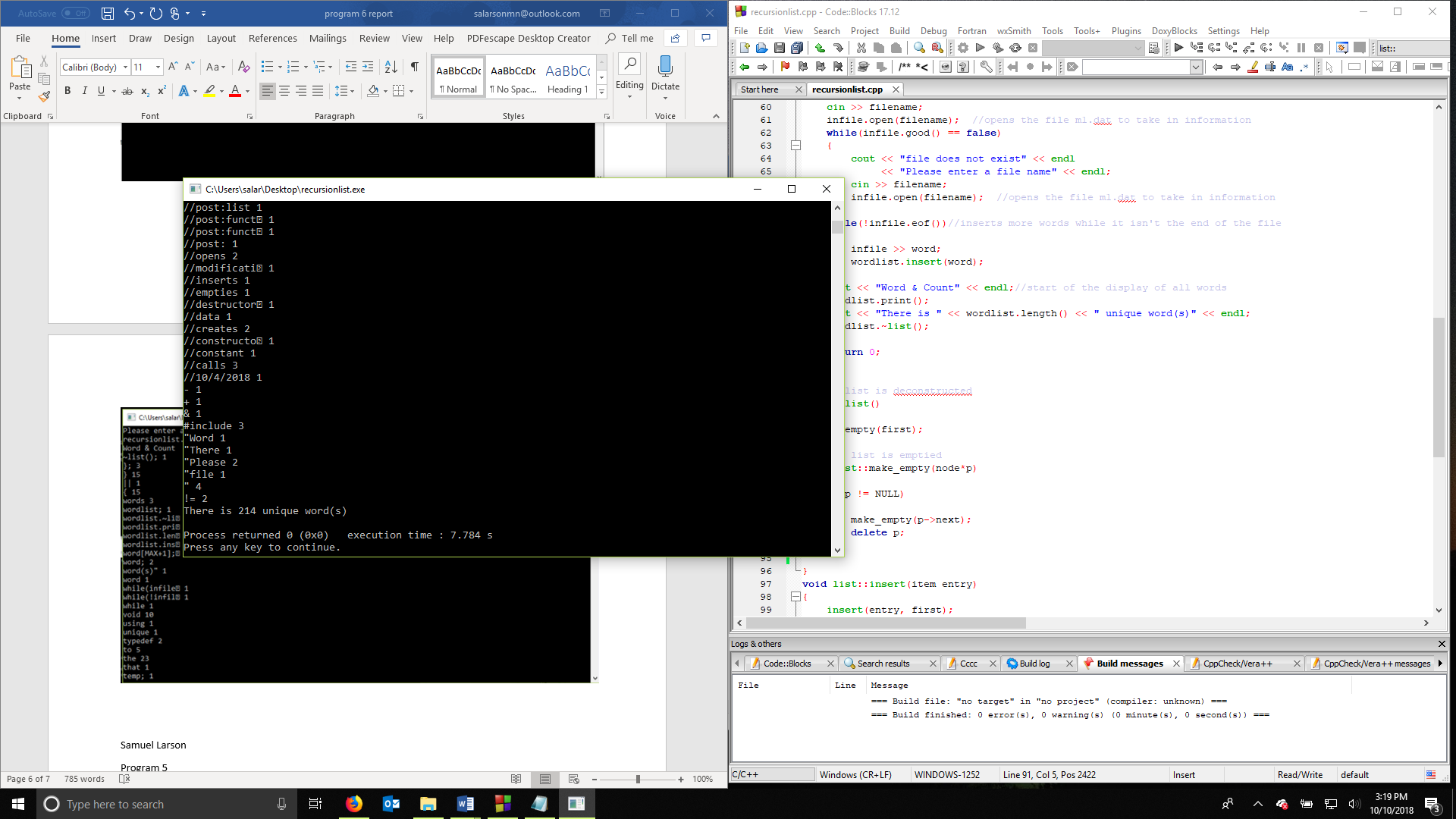
Program 5

10/11/18









Samuel Larson

Program 5

10/11/18

User Document

To use this program the program must be in the same directory as the file of text you would like to read. When you launch the program, it will prompt you to enter the file name and extension

Ex. Text.txt or wordlist.cpp

Hit enter and the program will proceed to track each unique word in the document and add them to a list with how many times they appear in the document.

The program will print the resulting information to the screen.

Samuel Larson

Program 5

10/11/18

Summary

The purpose of this project was to rethink how project 4 could be done using recursive functions the program had to read words in from a text file and add them to a list while keeping track of the count and find how many unique words there are.

This program could be improved by finding the big O of the functions in this program and comparing it to the big O of the functions in program 4 and using the method for the functions that is most efficient

More functions could be added to the program to give the user more options with the list. Saving the list, loading a list from file or removing and adding individual data to the list