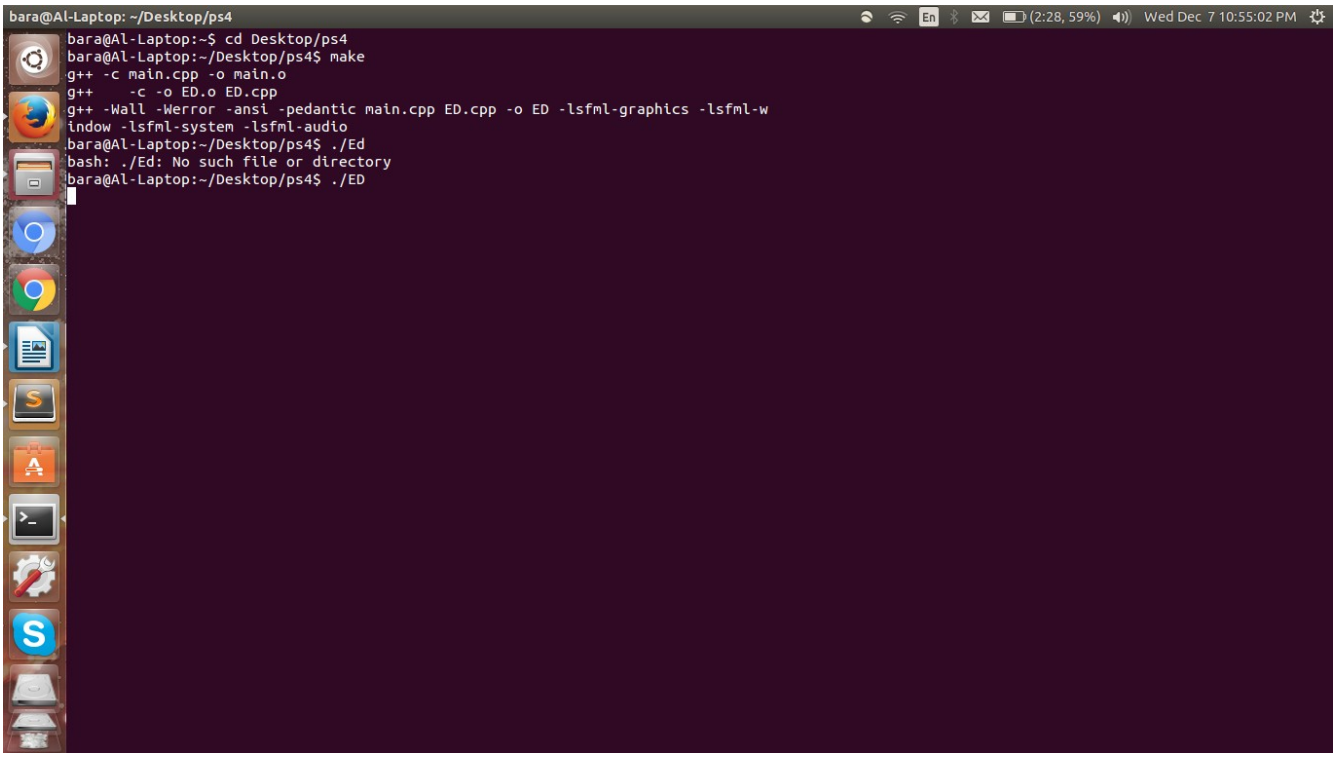
<u>PS4</u>: *Edit Distance*

In this assignment, this program is to create a optimal sequence alignment of two DNA strings. It will measure the similarity of two genetic sequences by the edit distance. It can be used for plagiarism detection, file revisioning. This is accomplished when two letters are aligned. If they are the same, it cost 0 bits. If they are different, it costs 1 bit. If their inset any at all, it will cost 2 bits.

This assignment could be approached in many different ways. Recursion and dynamic programming were recommend choices for this assignment. I tried taking the dynamic approach using an NxM matrix. Creating a class called ED, the constructor took two string were set to the member variables.

For the assignment, I didn't learn so much. However, I did learn the understand of edit distance. I learned what the necessary bits are needed, the alignment, and the total amount of distance needed for alignment.

Sadly, I was not able to finish this assignment. I believe I was confused on how to approach this assignment and felt lost on what to do next. When I tried running the program, I had a continues loop and had to close the the program by killing it.



```
cc = g++

all : main

main : main.o ED.o
        $(cc) -Wall -Werror -ansi -pedantic main.cpp ED.cpp -o ED -lsfml-graphics -lsfml-window -lsfml-system -lsfml-audio

main.o : ED.hpp
        $(cc) -c main.cpp -o main.o

ED : ED.cpp ED.hpp
        $(cc) -c ED.cpp -o ED.o

clean:
        rm *.o ED
```

```

#include "ED.hpp"

int main(int argc, char* argv[]){

        std::string store;
        std::string string_1;
        std::string string_2;
        std::string answer;

        sf::Clock clock;
        sf::Time t;

        std::cin >> store;
        string_1 = store;
        std::cin >> store;
        string_2 = store;

        ED temp(string_1,string_2);

        //displays the execution time
        t = clock.getElapsedTime();
        std::cout << t.asSeconds() << "seconds" << std::endl;



        return 0;
}
```

```
#ifndef ED_H
#define ED_H

#include <string>
#include <iostream>
#include <algorithm>
#include <SFML/System.hpp>

class ED{
private:
  std::string _str1;
  int _str1Len;
  std::string _str2;
  int _str2Len;

  int **_array;

public:
  ED(std::string str1, std::string str2);//constructor, allocates any data s
tructures

  static int penalty(char a, char b);//returns penaluty for aligning chars(0
 or 1)

  static int min(int a, int b, int c);//returns minimum of 3 args

  int optDistance();//populates matricies based on two strings, returns opti
mal discance

  std::string alignment();//traces the matrix and returns string to be print
ed

  int getStr1Len();

  int getStr2Len();

  ~ED();

};

#endif
```

```
#include <string>
#include "ED.hpp"

//constructor
ED::ED(std::string str1, std::string str2){
  //basic assignment
  _str1 = str1;
  _str2 = str2;

  _str1Len = str1.size();
  _str2Len = str2.size();

  //allocate for the first dimension
  _array = new int*[_str1Len + 1];//+1 because we assume the first column/row is empty

  //need to allocate 2nd dimension of array, <= because _array is (str1Len + 1

}

//get the penalty for comparing the args
int ED::penalty(char a, char b){

  //test = b - a;
  if(a == b)//check if chars are the same
    return 0;
  return 1;
}

//returns minimum of the three args, basic comparison
int ED::min(int a, int b, int c){
  if(a < b && a < c)
    return a;
  if(b < c)
    return b;
  else
    return c;
}

//traverses the 2d array, the meat of the program, have to move backwards (start at end)
/*int ED::optDistance(){
  //traverse from the end of the y-axis to the front

}

//traces the matrix and returns a string
std::string ED::alignment(){


}*/

int ED::getStr1Len(){
  return _str1Len;
}

int ED::getStr2Len(){
  return _str2Len;
}

```

```
ED::~ED(){
  //need to delete the memory we used for _array, this is second level
  //need to delete the first level
  delete[] _array;
}
```