COMP 215 Algorithms

Lab #6

No more bizarre tricks in C++ just for the sake of doing bizarre tricks in C++. From now on, we will be implementing algorithms seen in class (this doesn't mean however that we will stop doing bizarre ticks in C++, the tricks will still show up here and there as needed for implementing the algorithms).

In this lab, you will implement the brute force String Matching and Closest Pair algorithms. Submit your .cpp file on onCourse before Friday, October 9 at 11:59pm.

First, string matching. Not much preliminary work here since the string class in C++ already contains all the methods needed to implement the algorithm.

1. Write a function `StringMatchingBF` that takes two strings as input and returns the index of the first location in the first input string where the second input string is found. So you're pretty much just implementing the algorithm exactly as it is in the book, except that you don't need to get the length of each string as input, there's a function in C++ to find that from your input.

For the Closest Pair algorithm, we first need to make our own data structure.

2. Create a class `point2D` that contains two public data members: an x coordinate and a y coordinate, both integers.
3. Write a constructor that takes two integers as input and uses them to initialize the x and y coordinates of the point.
4. Write a `toString` function that takes no input, and outputs a string representing the point. If the x coordinate of the point is 11 and the y coordinate of the point is 5, the `toString` function should return the string `"(11,5)"`.

Now with the algorithm.

5. Write a function `ClosestPairBF` that takes as input an array of pointers to `point2D`, its size, and two other integers passed by reference that will be used to return the indices of the closest points. The function should not return anything (apart from the two indices passed by reference). The function should implement the function seen in class. Note that this is not the same as the function in the book, which returns the distance instead of the pair of points.

Finally, the main function.

6. First, your main function should ask the user for the name of an input file, and ask if that file is a text file, or a file of numbers.
7. Depending on the user's input, do the following:
   - If the input file is a text file, the first line of the file will contain the number of other lines in the file. Read the entire file and store the text in a single string. As you read the file, you should remove all the newline characters, and make sure that all the words are separated by a single space. Your program should then ask the user for strings to search in the text from file, and run the string matching function on these inputs. The program should keep doing this until the user enters an empty string. *Be careful!* the user input should be allowed to contain spaces, so you have to do the user input carefully.
   - If the file is a number file, the first line of the file will contain the number of other lines in the file. All subsequent lines contain two integers: the x and y coordinates of points in the plane. Read the entire file, store all the points in an array of pointers to `point2D`. Print on screen the two indices and the points (using the `toString` function) that are closest.

You will find on onCourse a .zip file that contains a few possible input files. Use them to test your program, or create your own to make additional tests. The numbers in the input files are small enough than calculating distances should not go out of the long integer format.

Want to do more? OK then.

BONUS:

8. In the class `point2D`, overload the operators ==, <, >, <= and >= such that $[(a, b) == (c, d)$ if and only if $a == c]$ and $[(a, b) < (c, d)$ if and only if $a < c]$ (so we compare only the x coordinate, and ignore the y coordinate for the comparison). The other operators should be the extension of these two.
9. Write a function `BubbleSort` that takes an array of pointers to `point2D` and its size. The function should implement the Bubble Sort algorithm, duh.
10. In the main, if the user decided to use a number file, after executing the closest pair function, use Bubblesort to sort your array and print the result on screen.