Lab 11: Searching and Sorting (60pts) Professor Conroy
CSC111 Spring 2020 2 April 2020

Your friend, the humble farmer, is very pleased with your work for the robot design to automate agriculture from Lab 10. They have requested that you add a few more features to the API of your robot so that it can scale to an industrial level.

The robot should be augmented by adding the following functions:

int plant(char, char[][10]) – plants a crop at all spaces in the farm where there is an empty plot using the character passed in as parameter as the plant of choice. So if the farm is something similar to:

0, 0, X, X,
p, R, b, 0,
0, 0, 0, 0

And you call plant('r', arr); The new plot should look like this:

r, r, X, X,
p, R, b, r,
r, r, r, r,

You should use the original plant() function to plant a single plant (from Lab 10) inside this new plant function to save yourself some work. You should return how many new plants were planted on the farm plot.

vector<char> harvest(char[][10]) – picks all plants that are ready to be eaten (those that are capital but not 0 or X). The character of the plants should be stored in a vector that will be returned by the function. At any location on the farm where a plant is harvested a '0' is placed. If there is no plant ready to be harvested at a location nothing needs to be done for that cell. You can use your original harvest function to harvest each individual plant inside this harvest function to save yourself some work. For example, the farm:

0, 0, X, X,		0, 0, X, X,
p, R, b, B,	should become -->	p, 0, b, 0,
0, 0, P, 0		0, 0, 0, 0

and return a vector containing {R,B,P}

int clean(char[][10]) – removes all X's from the farm and replaces the X's with a zero. The function should return a count of how many dead plants were removed. So for instance, the farm:

0, 0, X, X,		0, 0, 0, 0,
p, R, b, 0,	Should become -->	p, R, b, 0,
0, 0, 0, 0		0, 0, 0, 0

and return a 2 for having removed two dead plants. You can use your harvest function from the previous lab to make the code simpler to write.

void bubble_sort(vector<char>&) - takes any collection of plants and sorts them into alphabetical order using the bubble sort algorithm. Please consult your textbook for guidance on writing this.

void insertion_sort(vector<char>&) - takes any collection of plants and sorts them into alphabetical order using the insertion sort algorithm. Please consult your textbook for guidance on writing this.

So for instance, for either sort function, the plant collection:

{P, B, R, R, P, B, B, P} should become {B, B, B, P, P, P, R, R}

int main() - Your main function should be capable of reading a sequence of farms from a provided input file (data.txt) and calling your newly written clean(), plant() and harvest() functions on those farms. Your program should keep track of how many of each plant was planted and how many of each plant was harvested. All harvested plants should be stored in a single vector that is sorted at the end of processing all of the farms. You may choose to plant, harvest, and clean a farm in any order and you may choose to plant whatever crops you desire.

At the end of the program, you should write how many of each plant was planted and how many of each plant was harvested as information in a file. At the end of the file, you should also print out the contents of the final sorted vector whose contents represents all of the plants picked from all of the farms. Make sure the number of each plant in the final sorted vector agrees with your count of how many of each plant was harvested; in fact, you are welcome to use the final sorted vector as the data set from which you determine the final count of each harvested plant.

Your program should work for any file without pressuposing any knowledge of the file's contents except for the format (5 X 10 farms consisting of characters r, b, p, R, B, P, X, 0 with each farm separated by a space). See the file data.txt to get a sense of how it is structured and organized.

Feel free to write additional functions to help organize the work in main. For instance, you might consider writing a function that does the work to generate each multidimensional array representing a single farm from the data in the file. The way you organize your main function is entirely at your discretion. It might be helpful for you to get the required functions above working before you try to write your final main function.

Extra Credit (10pts): Write a function, **int binary_search(char, vector<char>)** that takes any sorted vector and determines the position of a particular plant. Please consult your textbook for guidance on writing this.

Extra Credit (10pts): Write your code so that it works with any size farm

Extra Credit (20pts): Create an additional output file that generates plant and harvest stats for each individual farm for each individual crop