Data Structure - Quadtree Assume the existence of a regular Cartesian grid that defines the 2D domain $[-1,1]^2$. In each direction of the x or y-axis, the interval $[-1,1]$ is discretized into N equal subintervals, leading to a total of $(N+1)^2$ grid points (x_i, y_j) where $0 \leq i, j \leq N$. Any four points (x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}) and (x_{i+1}, y_{j+1}) form a rectangular cell. In total there are N^2 cells. Suppose a scalar function is measured on the grid points with values f_{ij} at point (x_i, y_j).

Define a scalar function $f_{ij} \in \mathbb{R}$, for all $0 \leq i, j \leq N$.

Problem to solve: Given an arbitrary constant c, find the subset of cells of the Cartesian grid, such that c lies between the minimum and maximum of the function f over all four corners of each cell.

Develop an algorithm to solve this problem. The simplest approach is to do a search cell by cell. Thus the total complexity would be at least $O(N^2)$. We are not satisfied with this complexity and would like to find a better algorithm. So, you are required to:

1. solve the problem using the brute-force method via a cell by cell search.

2. construct a quadtree to store your data, such that for any given c, we can solve the problem with an asymptotic complexity less than $O(N^2)$.

3. Write some pseudo-code to preprocess your data into the quad-tree data structure.

4. Implement your pseudo-code in C++.

5. Write a pseudo-code for an algorithm to find the cells for any given c. And implement it in C++.

6. Apply your code to the following two functions

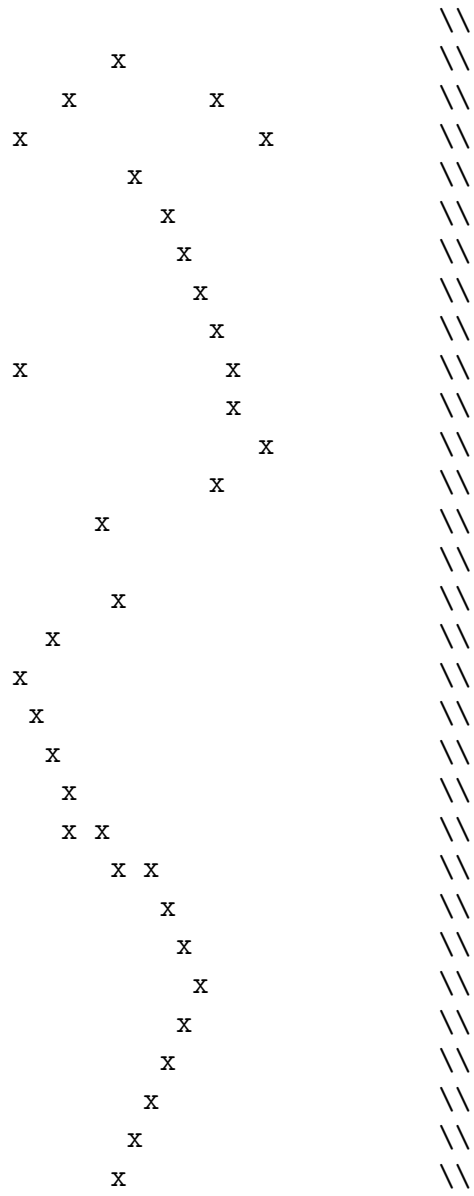
$$f_{ij} = f(x_i, y_j) = x_i^2 + y_j^2,$$

, and

$$f_{ij} = f(x_i, y_j) = x_i^2 - y_j^2.$$

Implement your two functions using polymorphism (similar to the previous homework). This makes it easy to add additional functions. (continued on next page)

Choose any c between 0 and 1. Output your results in the following tabular format with N rows and N columns:



where an 'x' is output for each cell found satisfying the condition. All

other cells will store the blank character ' '.

7. Run both versions of the algorithm (brute force and quad-tree) for N=32, 64, 128, 256, 512, 1000, and time the two versions for each value of N. Make a plot of time as a function of N for the two algorithms and overlay the two curves. The functions and value of c used are the same for the two algorithms.

8. use doubles for the function. No need to use templates. If you understand the Standard Template Library, feel free to use it. It is not a requirement.

9. Include a Makefile for the project.

10. Document your code and submit source code and documentation to canvas. Each function should be documented with the following

    (a) A one-line description of the purpose of the function

    (b) A brief explanation for each parameter and return value

Note: How you time the different sections of your code is up to you.