**CSE 2320-001 Lab Assignment 1**

Due February 14, 2012

**Goals:**

1. Application of sorting.

2. Application of binary search.

**Requirements:**

1. Design, code, and test a C program to perform *range queries* for two dimensional point data. The first line of the input will be *n*, the number of integer coordinate pairs in the next *n* input lines. *n* will not exceed 5000. Each of the remaining lines will have four values giving the coordinates of the lower left point and upper right point of a *bounding box*. The last input line will be `-1 -1 -1 -1`. All input point coordinates will be in the range `0 .. 1000`, inclusive. The input should be read from standard input (`stdin`). Your code should preprocess (sort) the points and use O(*m* + log *n*) time to process each query, where *m* is the the number of points in the smaller of the horizontal and vertical "slabs". For each query you should output 1) *m*, 2) the coordinates of the points in the answer, and 3) the number of points in the answer.

2. Email your program (as attachments) to `adnan.khan@mavs.uta.edu` by 1:45 pm on February 14. The `Subject` should be your name as recorded by the University and you should `cc:` yourself to verify that you sent the message correctly.

**Getting Started:**

1. It is easy to program this **in**efficiently. Simply read the points into arrays `x` and `y` and then for each bounding box (`xLL yLL xUR yUR`) apply the following test to each point `i`:

```
x[i]>=xLL && x[i]<=xUR && y[i]>=yLL && y[i]<=yUR
```

2. To make the query processing more efficient, a sorting-based preprocessing phase is used. At the end of preprocessing, there will be sorted tables of x-values and y-values, along with two tables storing the other coordinate. In the following example, tables `x` and `yPri` go together and so do tables `xPri` and `y`.

```
10              i    x    y xPri yPri
0 4             0    0    1    6    4
3 2             1    1    2    3    6
5 7             2    2    3    9    5
9 3             3    2    4    0    4
2 4             4    3    4    2    2
3 7             5    3    5    2    7
2 5             6    4    6    1    8
1 6             7    5    7    3    7
6 1             8    6    7    5    1
4 8             9    9    8    4    3
```

3. For processing each query, four binary searches are used to determine which pair of tables will be faster. For example, if the query is `2 1 5 2`, using table `x` will be slower than using table `y`.

4. Your program should not prompt for an input file name. Instead, a shell redirect (`a.out < file1.dat`) or a pipe (`cat file1.dat | a.out`) may be used to access data in a file.

5. Arrays should be allocated dynamically.

6. You may sort using any technique you would like, including the standard `qsort()`.