PDS Lab - Section 9

Lab Test 2

May 4th, 2020

Instructions: The lab test has 2 alternative questions. If you attempt the main problem, you will be evaluated for the full marks. If you are not able to attempt the main problem but attempt the alternative question, you will only be graded out of 50% of the marks of the lab test. [Thus if you get full marks for the alternative , that would be scaled down to 50% of the lab test marks.] You can not submit both the questions, if you do, only the second question will be considered.

You must use a single C files to write the program. The name of the file will be ⟨**RollNo**⟩**.c**.

Example: For the roll no. 18CS30012, the two programs should be named as: **18CS30012.c**

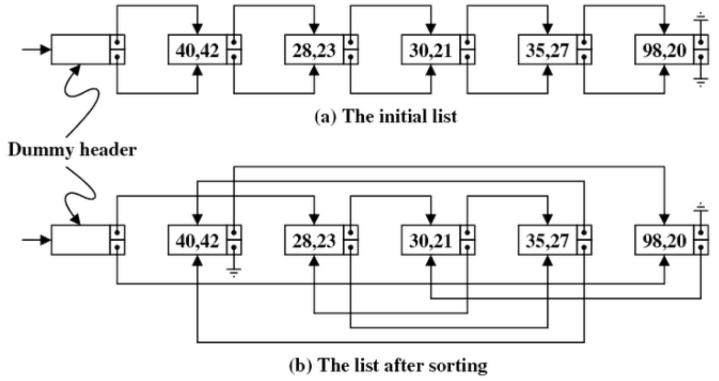
Your files should include your roll Number and name at the top. Also include whether you are doing the main or alternative question. For example:

/* Roll No. XXXXX Name: YYYY Main question*/

Main Question

Suppose that we have a linked list of points in the X-Y plane. We want to sort the list simultaneously with respect the X-coordinates and with respect to the Y-coordinates. To achieve this goal, we maintain two pointers in each node. One set of these pointers is used to sort the list with respect to the X-values, the other with respect to the Y-values. The following figure demonstrates this concept where the input consists of 5 points:

[(40,42), (28,23), (30,21), (35,27), (98,20)]



Note that in the list after sorting, the x- and y- co-ordinates of a point are not separated. Only the pointers are manipulated, and x- and y- pointers can point to different points. You can see that if you follow the x-pointers, you can get the X co-ordinates sorted and if you follow the y-pointers, you can get the y co-ordinates sorted.

The relevant datatype would be:

```
typedef struct _node {
      int x;
      int y;
      struct _node *nextx;
      struct _node *nexty;
   } node;
typedef node *list;
```

1. (6 marks) Read a list of 50 points with integer coordinates from the file 'points.txt'. At this point, you do not worry about sorting of the elements, but insert every new element at the end of the current list with respect to both the *nextx* and *nexty* pointers.

2. (6 marks) Write a function with the following prototype:

   ```
   void printList ( list L, int flag );
   ```

   This function prints the list headed by the pointer L. Now, each node has two pointers. The flag indicates whether the list is to be traversed along the *nextx* pointers, or along the *nexty* pointers. Your main should call this function to print the list.

3. (12 marks) Write two functions with the following prototypes:

   ```
   void bubbleSortX ( list L );
   void bubbleSortY ( list L );
   ```

   The first function is meant for bubble-sorting the list headed by L with respect to the x-values, and the second with respect to the y-values. **You are not supposed to separate the x and y coordinates of a point.**

4. (6 marks) Your main function should call these functions in sequence. Then, it should print the list i) by traversing along the *nextx* pointers and ii) by traversing along the *nexty* pointers.

   Thus, for the example in figure: output will look like

   ```
   Printing as per X co-ordinates: 28 30 35 40 98
   Printing as per Y co-ordinates: 20 21 23 27 42
   ```

Alternate Question:

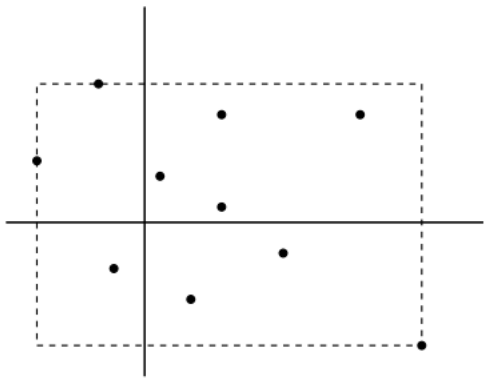
Randomly select 50 points given by X and Y coordinates. The coordinates are restricted to be integer values between -50 and +50 (both inclusive). Store the coordinates in two integer arrays.

1. (3 marks) Print the coordinates of these points using the array elements.

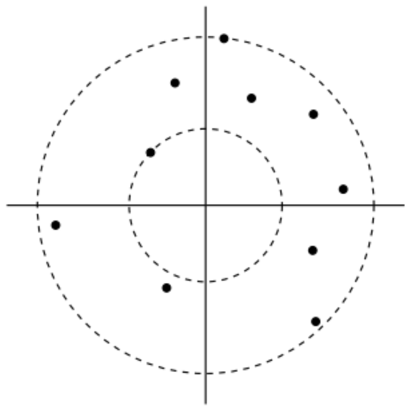
2. (3 makrs) Find the minimum and maximum of the X-coordinates of these points. Print these values.

3. (3 marks) Find the minimum and maximum of the Y-coordinates of these points. Print these values.

4. (6 marks) Determine the smallest rectangle with sides parallel to X and Y axes, that encloses all the given points. The coordinates of the corners of the rectangle and its area are to be printed. An example on 10 points is given below:



5. (4 marks) Now, compute the distance of each point fom the origin. Print these distances.
6. (4 marks) Find the minimum and maximum of these distances. Print these values.
7. (7 marks) Determine the smallest ring with center at the origin, that encloses all the given points. The inner and outer radii of the ring and its area are to be printed. An example on 10 points is given below:



Note that the printing should be neat. For each part, the printing should start with, for example

```
===Part 1====
The co-ordinates of 20 points are
(5,8)
...

===Part 2====
The minimum of the X-cordinates of tese points is: ___
....
```