INDIAN STATISTICAL INSTITUTE

MTech(CS) I year 2020-2021

Subject: Computing Laboratory

Lab Test 3 (5 March, 2021)

Total: 60 marks Duration: 3 hours

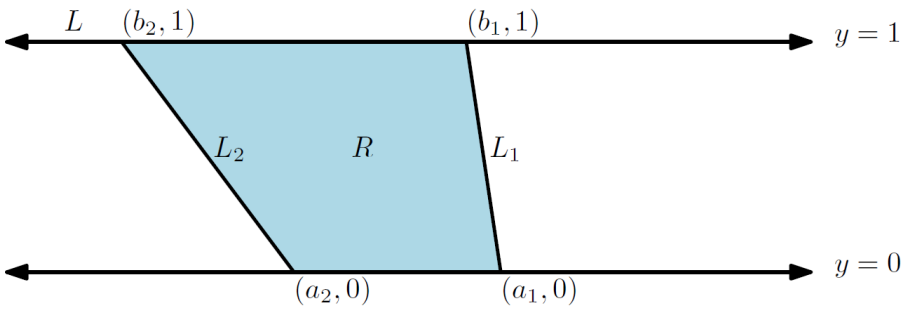
INSTRUCTIONS

1. You may consult or use slides / programs provided to you as course material, or programs that you have written yourself as part of classwork / homework for this course, but please **do not** consult or use material from other Internet sources, your classmates, or anyone else.
2. You may write your programs in C/C++ or Python.
3. Please make sure that your programs adhere strictly to the specified input and output format. **You may lose marks if your program violates the input and output requirements.**
4. The files `https://www.dropbox.com/s/441dlkf4g9k84cg/cs20XX-labtest3.py?dl=1` and `https://www.dropbox.com/s/stl8yjsyasy1vhl/cs20XX-labtest3.c?dl=1` contain a skeleton for the program that you have to write. Download the appropriate file, rename it by replacing `XX` with your 2-digit roll number, and modify the contents as required. You may define any additional variables/functions that you need in your program.

   PLEASE DO NOT CHANGE THE FILE NAME IN ANY OTHER WAY.
5. Please upload your program to `https://www.dropbox.com/request/NkhcdbbTYiqIBnRMXXEU`.

Q1. Recall that the equation of any straight line L in the X-Y plane can be written as $ax + by + c = 0$. Write a function `whichSideOfLine()` to determine whether a given point $P = (p, q)$ lies to the **left** or **right** of (or **on**) a given line L that is not parallel to the X-axis. Your function should take a, b, c, p, q as arguments, and return -1, 0, or +1 depending on whether P lies to the left of, on, or to the right of L, respectively. [5]

Q2. Consider a pair of non-intersecting (within $y = 0$ and $y = 1$) straight line segments L_1 and L_2, which are not parallel to the X-axis. Each of the segments L_1 and L_2 connects a point on the X-axis ($(a_1, 0)$ and $(a_2, 0)$ respectively) to a point on the line L given by $y = 1$ ($(b_1, 1)$ and $(b_2, 1)$ respectively). Let R denote the trapezium enclosed by L, L_1, L_2, and the X axis. The figure below shows an example of L_1, L_2 and R.



Given a point $P = (p, q)$ lying in the region between $y = 0$ and $y = 1$, you have to determine whether P lies to the left of, within, or to the right of region R. Write a function `whichSideOfRegion()` that takes a_1, b_1, a_2, b_2, p, q as arguments and returns -1, 0, or +1 depending on whether P lies to the left of, within, or to the right of R, respectively. You may assume that the given point P will **not** lie on the boundary of R. [10]

Q3. Now consider a set of n non-intersecting (within $y = 0$ and $y = 1$) straight line segments $L_1, L_2, \ldots, L_n$, which are not parallel to the X-axis. The line segment L_i connects a point $(a_i, 0)$ on the X axis to a point $(b_i, 1)$ on the line L given by $y = 1$. The region between the X axis and L is partitioned by these line segments into $n + 1$ regions. Let $R_1, R_2, \ldots, R_{n+1}$ denote these regions.

Figure 1 shows an example with 5 line segments. Each region R_i $(2 \leq i \leq 5)$ may be specified by (the indices of) its left and right bounding segments, i.e., if region R_i is bounded on the left and right by line segments L_l and L_r respectively, then we may denote R_i by the tuple representation (l, r).

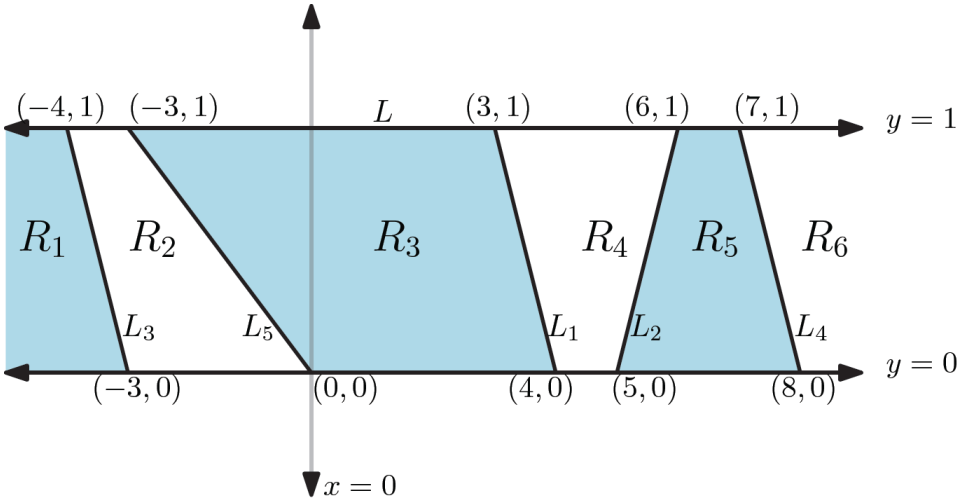


Figure 1: A scenario highlighted in Q3.

For example, the regions $R_2, \ldots, R_5$ (shaded alternately in white and blue) may be represented as $R_2 = (3, 5)$, $R_3 = (5, 1)$, $R_4 = (1, 2)$, and $R_5 = (2, 4)$. The first and the last regions (R_1 and R_{n+1}, respectively) are unbounded on one side. For the example in Figure 1, we represent them as $R_1 = (-1, 3)$ and $R_6 = (4, -1)$ respectively.

Note that the line segments are **not necessarily** numbered in increasing or decreasing order from left to right, but the regions **are** numbered (starting from 1) in ascending order from left to right.

(a) Write a function `findAndPrintRegions()` that takes the list $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ as arguments, and returns a list of the regions R_1, R_2, ... , R_{n+1} from left to right denoted using the tuple notation mentioned above. [15]

(b) Write a function `buildBST()` that builds and returns a binary search tree in which each node corresponds to a region R_i, and R_i is regarded as less than R_j iff R_i occurs to the left of R_j. Your tree should have the middle region R_m (where $m = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$) as its root. [20]

(c) Given a point $P = (p, q)$ lying in the region between $y = 0$ and $y = 1$, but not on any of the input line segments L_i, use the tree built above to find and print the region R_k to which P belongs. [10]

Input Format

Input will be provided via standard input. The input will consist of a single line comprising one positive integer n, followed by $2n+2$ floating point numbers, corresponding to $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ and p and q (the coordinates of point P) respectively.

Output Format

Output is to be printed on standard output. The `findAndPrintRegions()` function should print the list of regions in the form of tuples (l, r) and delimited by spaces in a single line. The main function (corresponding to question Q3.c) should print the region number as output in the next line. The `buildBST()` function should not print anything.

Sample Input 0

```
5 4 3 5 6 -3 -4 8 7 0 -3 0 0.5
```

Sample Output 0

```
(-1, 3) (3, 5) (5, 1) (1, 2) (2, 4) (4, -1)
3
```

Sample Input 1

```
3 4 3 0 -3 -3 -4 -2.5 0.1
```

Sample Output 1

```
(-1, 3) (3, 2) (2, 1) (1, -1)
2
```

Sample Input 2

```
2 -2 -2 2 2 0.5 0.5
```

Sample Output 2

```
(-1, 1) (1, 2) (2, -1)
2
```