Problem G: Line & Circle Maze

Source file: `maze.{c, cpp, java}`
Input file: `maze.in`

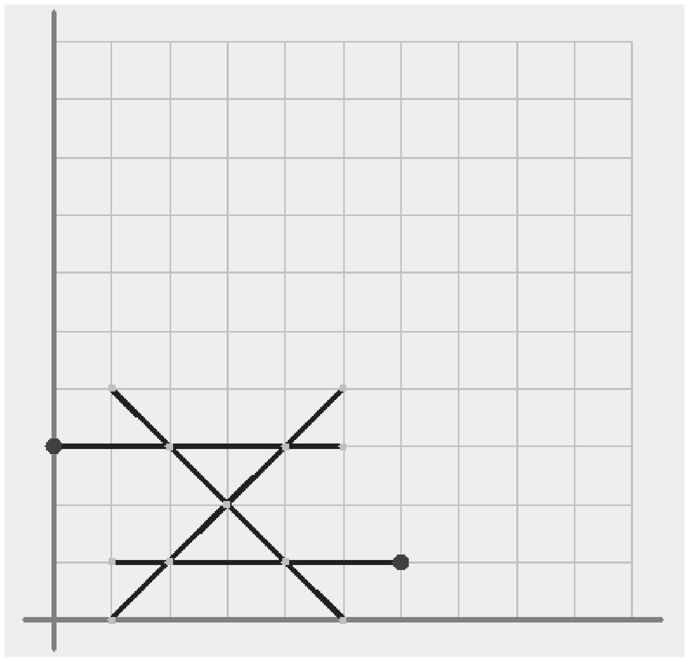
A deranged algorithms professor has devised a terrible final exam: he throws his students into a strange maze formed entirely of linear and circular paths, with line segment endpoints and object intersections forming the junctions of the maze. The professor gives his students a map of the maze and a fixed amount of time to find the exit before he floods the maze with xerobiton particles, causing anyone still in the maze to be immediately inverted at the quantum level. Students who escape pass the course; those who don't are trapped forever in a parallel universe where the grass is blue and the sky is green.

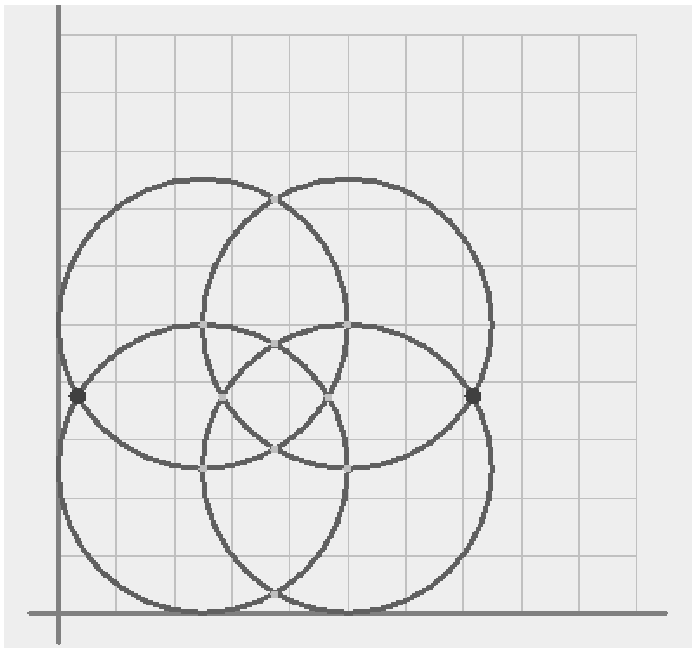
The entrance and the exit are always at a junction as defined above. Knowing that clever ACM programming students will always follow the shortest possible path between two junctions, he chooses the entrance and exit junctions so that the distance that they have to travel is as far as possible. That is, he examines all pairs of junctions that have a path between them, and selects a pair of junctions whose shortest path distance is the longest possible for the maze (which he rebuilds every semester, of course, as the motivation to cheat on this exam is very high).

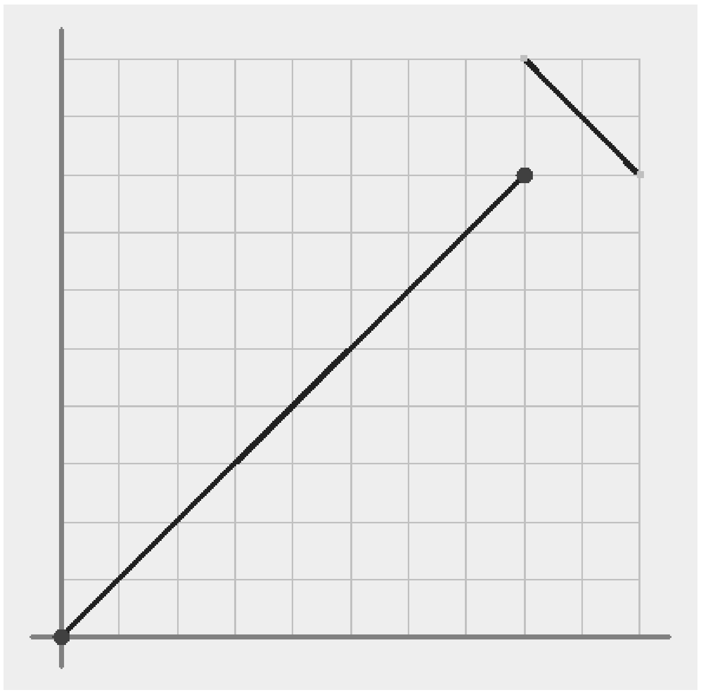
The joy he derives from quantumly inverting the majority of his students is marred by the tedium of computing the length of the longest of the shortest paths (he needs this to know to decide how much time to put on the clock), so he wants you to write a program to do it for him. He already has a program that generates the mazes, essentially just a random collection of line segments and circles. Your job is to take that collection of line segments and circles, determine the shortest paths between all the distinct pairs of junctions, and report the length of the longest one.

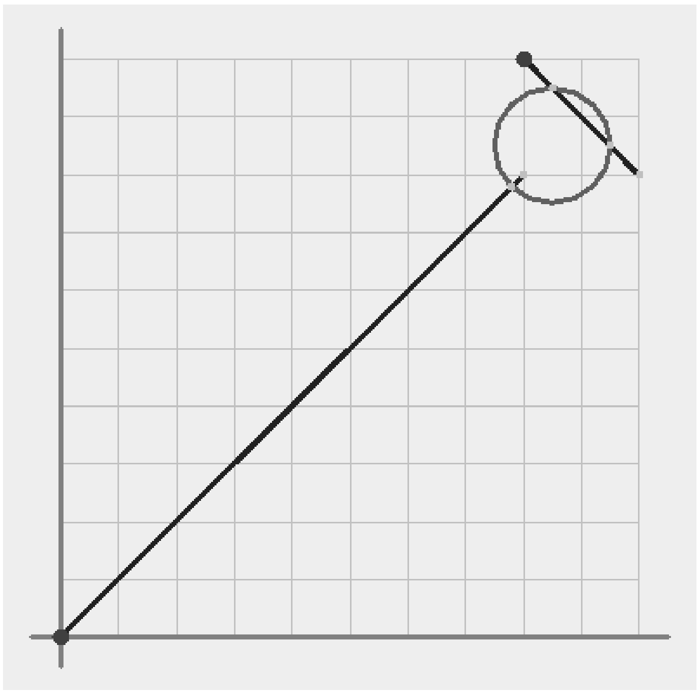
The input to your program is the output of the program that generates his mazes. That program was written by another student, much like yourself, and it meets a few of the professor's specifications: 1) No endpoint of a line segment will lie on a circle; 2) No line segment will intersect a circle at a tangent; 3) If two circles intersect, they intersect at exactly two distinct points; 4) Every maze contains at least two junctions; that is, a minimum maze is either a single line segment, or two circles that intersect. There is, however, one bug in the program. (He would like to have it fixed, but unfortunately the student who wrote the code never gave him the source, and is now forever trapped in a parallel universe.) That bug is that the maze is not always entirely connected. There might be line segments or circles, or both, off by themselves that intersect nothing, or even little "submazes" composed of intersecting line segments and circles that as a whole are not connected to the rest of the maze. The professor insists that your solution account for this! The length that you report must be for a path between connected junctions!

Examples:


Line segments only. The large dots are the junction pair whose shortest path is the longest possible.


An example using circles only. Note that in this case there is also another pair of junctions with the same length longest possible shortest path.


Disconnected components.


Now the line segments are connected by a circle, allowing for a longer shortest path.

Input: An input test case is a collection of line segments and circles. A line segment is specified as "L X_1 Y_1 X_2 Y_2" where "L" is a literal character, and (X_1,Y_1) and (X_2,Y_2) are the line segment endpoints. A circle is specified by "C X Y R" where "C" is a literal character, (X,Y) is the center of the circle, and R is its radius. All input values are integers, and line segment and circle objects are entirely contained in the first quadrant within the box defined by (0,0) at the lower left and (100,100) at the upper right. Each test case will consist of from 1 to 20 objects, terminated by a line containing only a single asterisk. Following the final test case, a line containing only a single asterisk marks the end of the input.

Output: For each input maze, output "Case N: ", where N is the input case number starting at one (1), followed by the length, rounded to one decimal, of the longest possible shortest path between a pair of connected junctions.

Example Input:	**Example Output:**

`L 10 0 50 40`
`L 10 40 50 0`
`L 10 10 60 10`
`L 0 30 50 30`
`*`
`C 25 25 25`
`C 50 25 25`
`C 25 50 25`
`C 50 50 25`
`*`
`L 0 0 80 80`
`L 80 100 100 80`
`*`
`L 0 0 80 80`
`L 80 100 100 80`
`C 85 85 10`
`*`
`*`
	`Case 1: 68.3`
	`Case 2: 78.5`
	`Case 3: 113.1`
	`Case 4: 140.8`

Note: It is recommended that the atan2() function, rather than acos() or asin(), be used when calculating arc angles.

Last modified on October 26, 2008 at 2:15 PM.