Programming Problem Instructions

Figure 1 shows a telescoping robot. In the code, you are given a set of **x,y** coordinates with corresponding pen up/down positions that will be used to setup moves for the robot. This robot has no rotation limits, but it does have limits on its reach (**RMIN** and **RMAX**). As described below, you will print a table of original movement data. You will then, **ONLY IF NECESSARY**, adjust any **x,y** coordinates that are outside the robot's reach, and reprint the table with the adjusted data. Finally, you will compose command strings for each move and print them out.

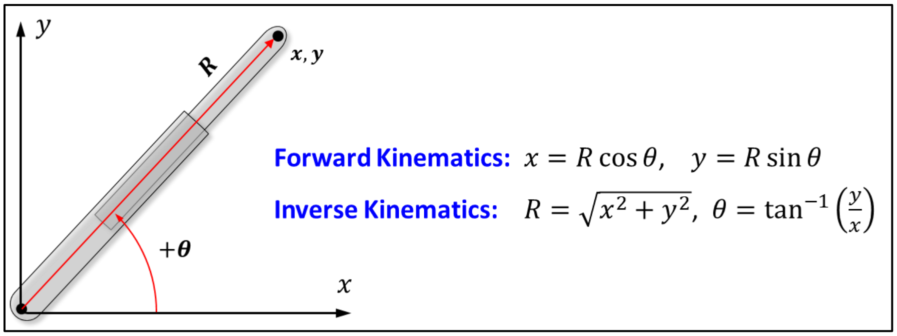
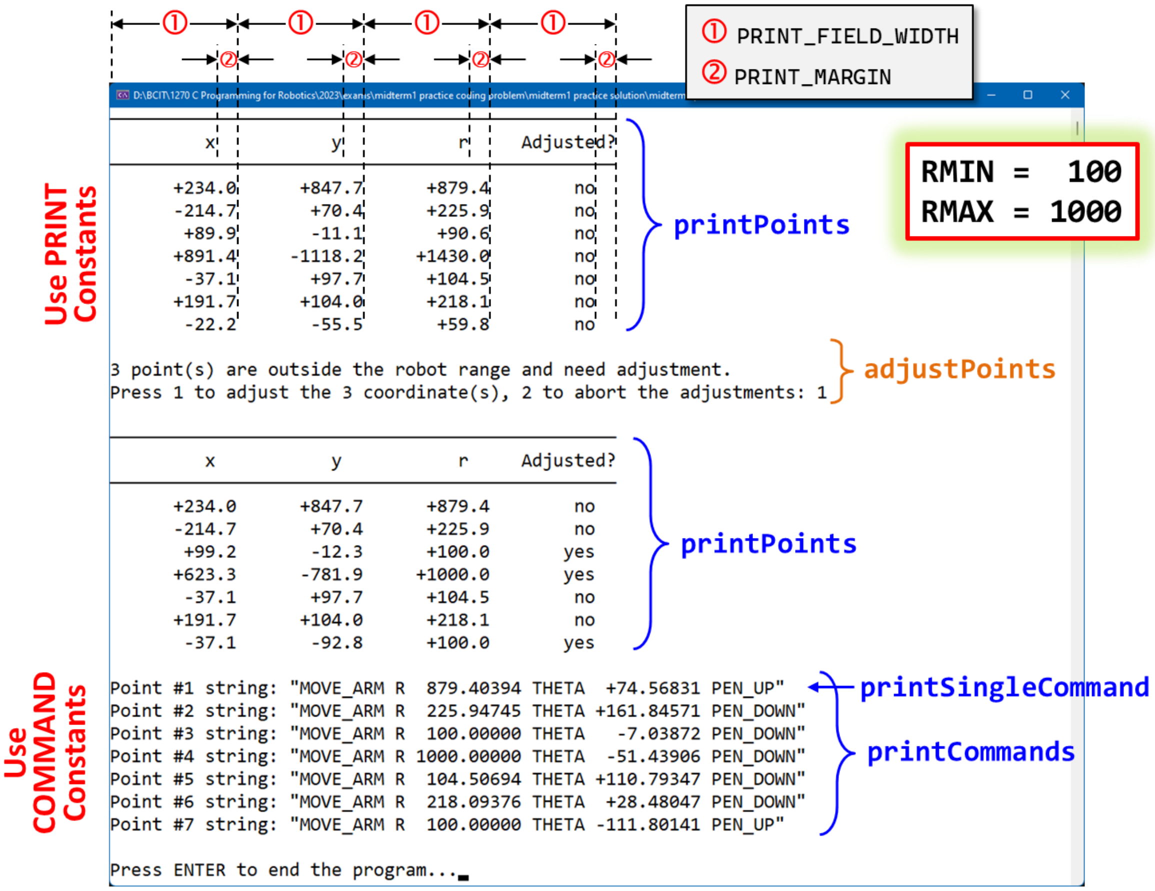

y
R
x, y
$+\theta$
x
Forward Kinematics: $x = R\cos\theta, \quad y = R\sin\theta$
Inverse Kinematics: $R = \sqrt{x^2 + y^2}, \; \theta = \tan^{-1}\left(\frac{y}{x}\right)$

FIGURE 1: TELESCOPING ROBOT

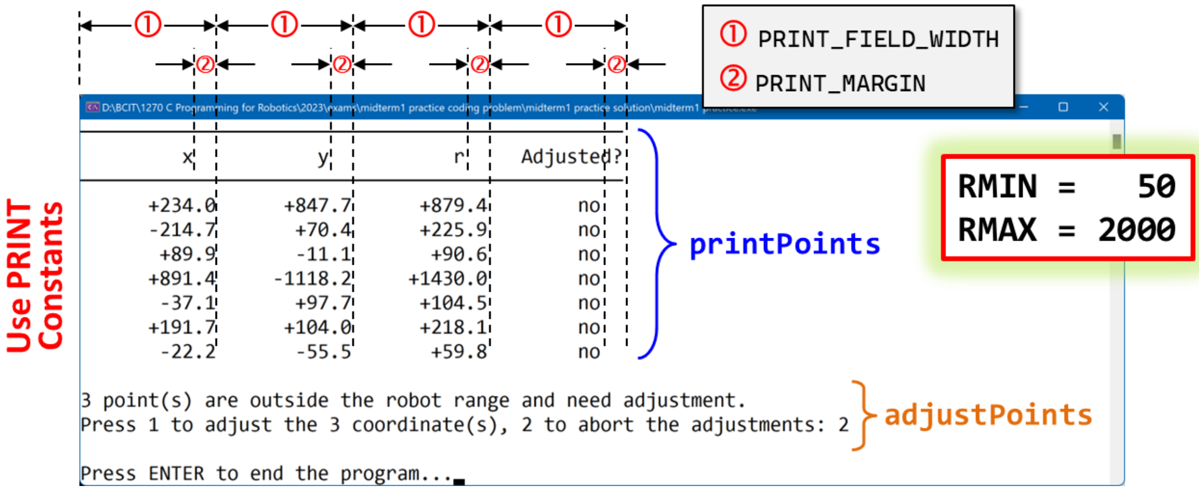
Sample outputs from the program are shown in Figures 2 and 3 (they are separate runs). **NOTE**: the examples use **the same** array values than you have in your code. Figure 2 shows the output when several points are outside the robot's reach.



```
────────────────────────────────────────────────
         x           y           r   Adjusted?
────────────────────────────────────────────────
      +234.0      +847.7      +879.4          no
      -214.7       +70.4      +225.9          no
       +89.9       -11.1       +90.6          no
      +891.4     -1118.2     +1430.0          no
       -37.1       +97.7      +104.5          no
      +191.7      +104.0      +218.1          no
       -22.2       -55.5       +59.8          no

3 point(s) are outside the robot range and need adjustment.
Press 1 to adjust the 3 coordinate(s), 2 to abort the adjustments: 1

────────────────────────────────────────────────
         x           y           r   Adjusted?
────────────────────────────────────────────────
      +234.0      +847.7      +879.4          no
      -214.7       +70.4      +225.9          no
       +99.2       -12.3      +100.0         yes
      +623.3      -781.9     +1000.0         yes
       -37.1       +97.7      +104.5          no
      +191.7      +104.0      +218.1          no
       -37.1       -92.8      +100.0         yes

Point #1 string: "MOVE_ARM R  879.40394 THETA  +74.56831 PEN_UP"
Point #2 string: "MOVE_ARM R  225.94745 THETA +161.84571 PEN_DOWN"
Point #3 string: "MOVE_ARM R  100.00000 THETA   -7.03872 PEN_DOWN"
Point #4 string: "MOVE_ARM R 1000.00000 THETA  -51.43906 PEN_DOWN"
Point #5 string: "MOVE_ARM R  104.50694 THETA +110.79347 PEN_DOWN"
Point #6 string: "MOVE_ARM R  218.09376 THETA  +28.48047 PEN_DOWN"
Point #7 string: "MOVE_ARM R  100.00000 THETA -111.80141 PEN_UP"

Press ENTER to end the program...
```

FIGURE 2: SAMPLE PROGRAM OUTPUT. SEVERAL POINTS OUTSIDE ROBOT RANGE

Figure 3 shows the output when adjustments are aborted.



①
②
① PRINT_FIELD_WIDTH
② PRINT_MARGIN
Use PRINT Constants
D:\BCIT\1270 C Programming for Robotics\2023\exams\midterm1 practice coding problem\midterm1 practice solution\midterm1 practice.exe
printPoints
adjustPoints
RMIN = 50
RMAX = 2000

```
        x         y         r Adjusted?

   +234.0    +847.7    +879.4        no
   -214.7     +70.4    +225.9        no
    +89.9     -11.1     +90.6        no
   +891.4   -1118.2   +1430.0        no
    -37.1     +97.7    +104.5        no
   +191.7    +104.0    +218.1        no
    -22.2     -55.5     +59.8        no

3 point(s) are outside the robot range and need adjustment.
Press 1 to adjust the 3 coordinate(s), 2 to abort the adjustments: 2

Press ENTER to end the program...
```

FIGURE 3: SAMPLE PROGRAM OUTPUT. ADJUSTMENTS ABORTED

To complete the code, you will complete **main (4 marks)** plus create and call the following **5 functions** (you can create more helper functions if you like):

1) **printPoints (10 marks)**
2) **adjustPoints (13 marks)**
3) **printCommands (3 marks)**
4) **printSingleCommand (5 marks)**
5) **inverseKinematics (5 marks)**

Detailed instructions for each function are given below.

main

Call printPoints once to generate the first table. Call adjustpoints and if it returns true, then call printPoints a second time and call printCommands. If adjustpoints returns false, end the program.

printPoints

This function prints a table of values stored in the **x** and **y** arrays. It also computes the radius value and "**no**" if the value in **bAdjusted** is false, "**yes**" if the value is true. Spacings and margins are shown in Figures 2 and 3 - use the global constant values in your code to line up the values. The field width and precision for the values are given by the global constants PRINT_FIELD_WIDTH and PRINT_PRECISION.

adjustPoints

This function adjusts any **x**, **y** pair that is outside the robot's reach (**<RMIN** or **>RMAX**). You will need to run through **main**'s **x**, **y** pairs and determine which need adjustment set **main**'s **bAdjusted** array element to **true** if so. You also need to keep a running count of how main need adjustment.

If no points need adjusting, return **false**.

If points need adjusting, follow the examples to tell the user how many points need adjustment and ask if they want to (make sure they give you good clean input data). If they choose to abort, return **false**. If they chose to proceed, the adjustments are done in two ways:

1. If **r<RMIN**, adjust **x** and **y** so that **r=RMIN** AND the angle ***θ*** is preserved (doesn't change from what it was originally).
2. If **r>RMAX**, adjust **x** and **y** so that **r=RMAX** AND the angle ***θ*** is preserved (doesn't change from what it was originally).

When you are finished adjusting point, return **true**.

printCommands

The function calls **printSingleCommand** for each row in the bottom section of Figure 2.

printSingleCommand

This function prints out a command string for a single move (**r**, **theta**, pen position). **r** and **theta** are calculated using the **inverseKinematicsFunction** (see below). Use the global constants **COMMAND_FIELD_WIDTH** and **COMMAND_PRECISION**. Example strings are:

```
MOVE_ARM R  879.40394 THETA  +74.56831 PEN_UP
MOVE_ARM R 1000.00000 THETA -111.80141 PEN_DOWN
```

Notice that the decimals line up in the same columns.

inverseKinematics

This bool function computes **r** and **theta** values from given **x** and **y** values. It returns **true** if the robot can reach the point, **false** if not. You must use pointers to get the **r** and **theta** values back to the calling function.

Hint: You can also use **inverseKinematics** in **adjustPoints**.