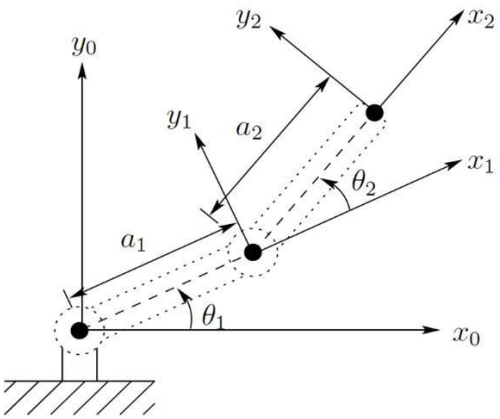
Saad Teeti

Problem Identification Statement

Create a software solution that performs complete analysis for the two-link planar robot. In order to do the analysis for the two-link planer, the software should involve performing five operations in order to analyze the two-link planar robot mechanism: double link mechanism, homogeneous transformation, DH convention, forward kinematics, and inverse kinematics. The user should input the length of the first and second link and a desired x, y coordinate, and the program should give him the two angels required to reach desired coordinate and a figure displaying the mechanism configuration.

Gathering Information

We use Inverse Kinematics equations, forward kinematics equations, Homogeneous transformation equations, and other link equations to write this code.



Inverse Kinematics Equations: To find the angles in order to move the hand to the desired coordinates.

$$\cos\theta_2 = \frac{x^2 + y^2 - \alpha_1^2 - \alpha_2^2}{2\alpha_1\alpha_2} := D$$

$$\theta_2 = \tan^{-1}\frac{\pm\sqrt{1 - D^2}}{D}$$

Forward Kinematics Equations: To check which of the two sets of angles is correct.

$$x = a_1 c_1 + a_2 c_{12}$$

$$y = a_1 s_1 + a_2 s_{12}$$

Transformation: To calculate the transformation matrix for a DH set.

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1 c_1 \\ s_1 & c_1 & 0 & a_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We use the following equation to determine initial position of both link 1 and 2.

Link1 = [-a1, 0;

0, 0;

0, 0;

1, 1]

Link2 = [-a2, 0;

0, 0;

0, 0;

1, 1]

To find the final position of the arm we use

*Link1 = A1 * Link1*

*Link2 = A1 * A2 * Link2*

Input/Output Diagram

Test Cases and Algorithm:

Test Cases1: valid input

>> Main

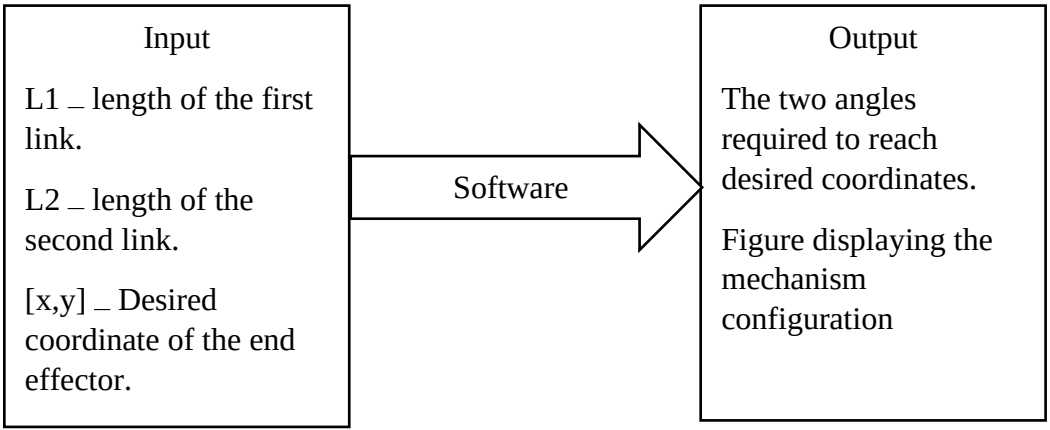
Enter the length of the first arm4

Enter the length of the second arm5

Enter the x coordinate of the end point of the robotic arm4 Enter

the y coordinate of the end point of the robotic arm6

angle1 =



74.0380

angle2 =

14.5015

Test Case 2: Valid input

>> Main

Enter the length of the first arm 7

Enter the length of the second arm 8

Enter the x coordinate of the end point of the robotic arm 6

Enter the y coordinate of the end point of the robotic arm 9

angle1 =

87.9533

angle2 =

8.6527

>>

Test Case 3: Invalid Length of arm one

>> Main

Enter the length of the first arm -5 invalid

input. Enter the length of the first arm

Test Case 4: Invalid Length of arm two

>> Main

Enter the length of the first arm 5 Enter the

length of the second arm -7

invalid input. Enter the length of the second arm

Test Case 5: invalid desired coordinate point

>> Main

Enter the length of the first arm *3*

Enter the length of the second arm *3*

Enter the x coordinate of the end point of the robotic arm *4*

Enter the y coordinate of the end point of the robotic arm *4*

Invalid coordinate. Enter the x coordinate of the end point of the robotic arm *5*

Enter the y coordinate of the end point of the robotic arm: 1

Invalid coordinate. Enter the x coordinate of the end point of the robotic arm

Algorithm

-Main Function:

Call function: Get_Input

Call function: Inverse_Kinematics

Call function: Forward_Kinematics

Print angle1

Print angle2

Call function: Trans_Matrix

Call function: Graph

-Get_Input function

Print "Enter the length of the first arm"

Read value into L1

Repeat

If L1 is less or equal to zero,

Print "invalid input"

Print "Enter the length of the first arm"

Read value into L1

Print "Enter the length of the second arm"

Read value into L2

Repeat

If L2 is less or equal to zero,

Print invalid input

Print "Enter the length of the second arm"

Read value into L2

Print " Enter the x coordinate of the end point of the robotic arm"

Read value into x

Print "Enter the y coordinate of the end point of the robotic arm"

Read value into y

Repeat

If L1+L2 is smaller than the square root of x squared + y squared
Print "invalid Input"
Print " Enter the x coordinate of the end point of the robotic arm"
Read value into x
Print "Enter the y coordinate of the end point of the robotic arm"
Read value into y

Repeat

If L1-L2 is bigger than the square root of x squared + y squared
Print "invalid Input"
Print " Enter the x coordinate of the end point of the robotic arm"
Read value into x
Print "Enter the y coordinate of the end point of the robotic arm"
Read value into y

Inverse_Kinematics function

Declare theta1a,theta1b,theta2a,theta2b as floats. Declare

*D and assign (x^2+y^2-L1^2-L2^2)/(2*L1*L2) to it.*

theta1a= $\tan^{-1}$(square root of (1-D^2)/D) theta2a=

$\tan^{-1}$(-square root of (1-D^2)/D)

*theta1b= tan^{-1} (y/x)- tan^{-1} ((L2*sin(theta1a))/(L1+L2*cos(theta1a));*

*theta2b== tan^{-1} (y/x)- tan^{-1} ((L2*sin(theta2a))/(L1+L2*cos(theta2a));*

***Forward_Kinematics function** Declare*

xa, ya, xb, yb as floats.

*xa=L1*cos(theta1b)+L2*cos(theta1b+theta1a);*

*ya=L1*sin(theta1b)+L2*sin(theta1b+theta1a);*

*xb=L1*cos(theta2b)+L2*cos(theta2b+theta2a);*

*yb=L1*sin(theta2b)+L2*sin(theta2b+theta2a);*

if xa is smaller or equal to x+0.001 and xa is bigger or equal than x -0.001 ***and*** *if ya is smaller or equal to y+0.001 and ya is bigger or equal to y-0.001*

assign theta1a to angle1
assign theta1b to angle2

else if xb is smaller or equal to x+0.001 and xb is bigger or equal than x -0.001 ***and*** *if yb is smaller or equal to y+0.001 and yb is bigger or equal to y-0.001*

assign theta2a to angle1
assign theta2b to angle2
else print " Error, the position cant be reached"

Trans_Matrix function

Declare two 4 by 4 matrices and name each one of them array1 and array2.
array1 should contian
*cos(angle2), -sin(angle2), 0, L1*cos(angle2);*

*sin(angle2), cos(angle2), 0, L1*sin(angle2); 0, 0, 1, 0;*

0, 0, 0, 1]

Array2 should contain

*cos(angle1), -sin(angle1), 0, L2*cos(angle1);*

*sin(angle1), cos(angle1), 0, L2*sin(angle1);*

0, 0, 1, 0;

0, 0, 0, 1

Declare two 4 by 2 matrices and name each one of them link11 and link22

Link11 should contain

-L1, 0;

0, 0;

0, 0;

1, 1

Link 22 should contain

-L2, 0;

0, 0;

0, 0;

1, 1

*Declare link1 and assign array1*link11 to it*

*Declare link2 and assign array1*array2*link22 to it.*

Graph Function

Plot both link 1 and 2 on a graph.

Code or Implementation

```
% calling the get input function to ask user for the length
of arm one and
%two and the coordinates.
[L1,L2,x,y]=Get_Input();
% calling inverse kinematics function inorder to To find the
angles in
%order to move the hand to the desired coordinates.
[theta1a,theta1b,theta2a,theta2b]=Inverse_Kinematics(x,y,L1
,L2);
% calling forward kinematics function to check which of the
two sets of
%angles is correct.
[angle1,angle2]=Forward_Kinematics(x,y,L1,L2,theta1a,theta1
b,theta2a,theta2b);
% displaying angles on the screen
display(angle1); display(angle2);
% calling trans matrix function to calculate the
transformation matrix for
%a DH set
[link1,link2]=Trans_Matrix(L1,L2,angle1,angle2);
% calling graph function to display the graph
[figure]=Graph(link1,link2);


function[L1,L2,x,y]=Get_Input()
```

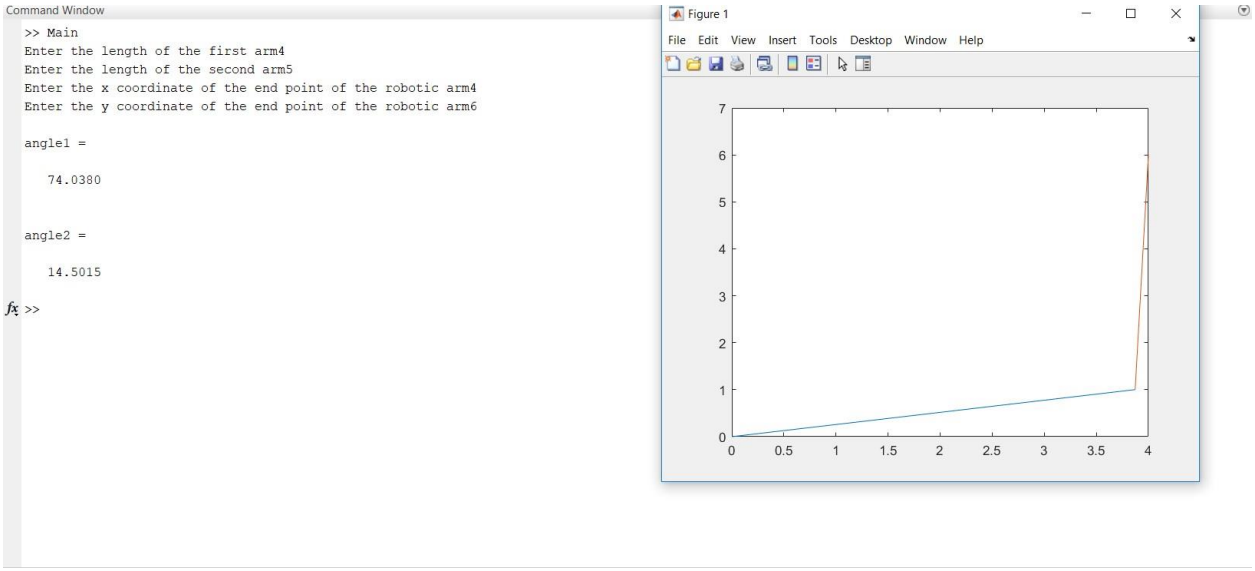
```
%asking the user to input value of the first arm and
checking if the value
%is valid
%if the value is not valid, the program will show and
invalid message and
%will ask the user to input the value again
L1=input('Enter the length of the first arm'); while
L1<=0
    fprintf('invalid input. ')
    L1=input('Enter the length of the first arm'); end
%asking the user to input value of the second arm and
checking if the value
%is valid
%if the value is not valid, the program will show and
invalid message and
%will ask the user to input the value again
L2=input('Enter the length of the second arm'); while
L2<=0
    fprintf('invalid input. ')
    L2=input('Enter the length of the second arm');
end
%asking the user to input value of the coordinates and
checking if the
%value is valid
%if the value is not valid, the program will show and
invalid message and
%will ask the user to input the value again
x=input('Enter the x coordinate of the end point of the
robotic arm'); y=input('Enter the y coordinate of the
end point of the robotic arm');
while(L1+L2<sqrt((x^2)+(y^2)))
fprintf('Invalid coordinate. ');
x=input('Enter the x coordinate of the end point of
the robotic arm ');              y=input('Enter the y
coordinate of the end point of the robotic arm: ');
end
while(L1-L2>sqrt((x^2)+(y^2)))
fprintf('Invalid coordinate. ');
x=input('Enter the x coordinate of the end point of
the robotic arm ');
            y=input('Enter the y coordinate of the end
point of the robotic arm: '); end  end
```

```
function[theta1a,theta1b,theta2a,theta2b]=Inverse_Kinematic
s(x,y,L1,L2)
% calculating d and inverse kinematics' equation to check
for two possible
%solutions for angles
D=(x^2+y^2-L1^2-L2^2)/(2*L1*L2);
theta1a=atand((sqrt(1-D^2))/D); theta2a=atand((-
sqrt(1-D^2))/D); theta1b=atand(y/x)-
atand((L2*sind(theta1a))/(L1+L2*cosd(theta1a)));
theta2b=atand(y/x)-
atand((L2*sind(theta2a))/(L1+L2*cosd(theta2a)));
end
function[angle1,angle2]=Forward_Kinematics(x,y,L1,L2,theta1
a,theta1b,theta2a,theta2b)
 %using forward kinematics equations to verify two possible
angles and
 %calculate x and y for both pairs of the angles.
xa=L1*cosd(theta1b)+L2*cosd(theta1b+theta1a);
ya=L1*sind(theta1b)+L2*sind(theta1b+theta1a);
xb=L1*cosd(theta2b)+L2*cosd(theta2b+theta2a);
yb=L1*sind(theta2b)+L2*sind(theta2b+theta2a);
%checking if x and y in each pair are equal to the desired
points through
%if statement and ifelse statements.
if (xa<=x+0.001 && xa>=x-0.001) && (ya<=y+0.001 && ya>=y-
0.001)     angle1=theta1a;     angle2=theta1b;
elseif (xb<=x+0.001 && xb>=x-0.001)&& (yb<=y+0.001 &&
yb>=y-0.001)     angle1=theta2a;     angle2=theta2b;
else
   fprintf('Error, the position cant be reached')
end end
 function[link1,link2]=Trans_Matrix(L1,L2,angle1,angle2)
%creating four by four array array1=[cosd(angle2), -
sind(angle2), 0, L1*cosd(angle2); sind(angle2),
cosd(angle2), 0, L1*sind(angle2); 0, 0, 1, 0;
0, 0, 0, 1]; array2=[cosd(angle1), -sind(angle1), 0,
L2*cosd(angle1); sind(angle1), cosd(angle1), 0,
L2*sind(angle1); 0, 0, 1, 0;
0, 0, 0, 1];
%creating two by four array
link11=[-L1, 0; 0, 0; 0, 0; 1, 1];
link22=[-L2, 0; 0, 0; 0, 0; 1, 1];
```

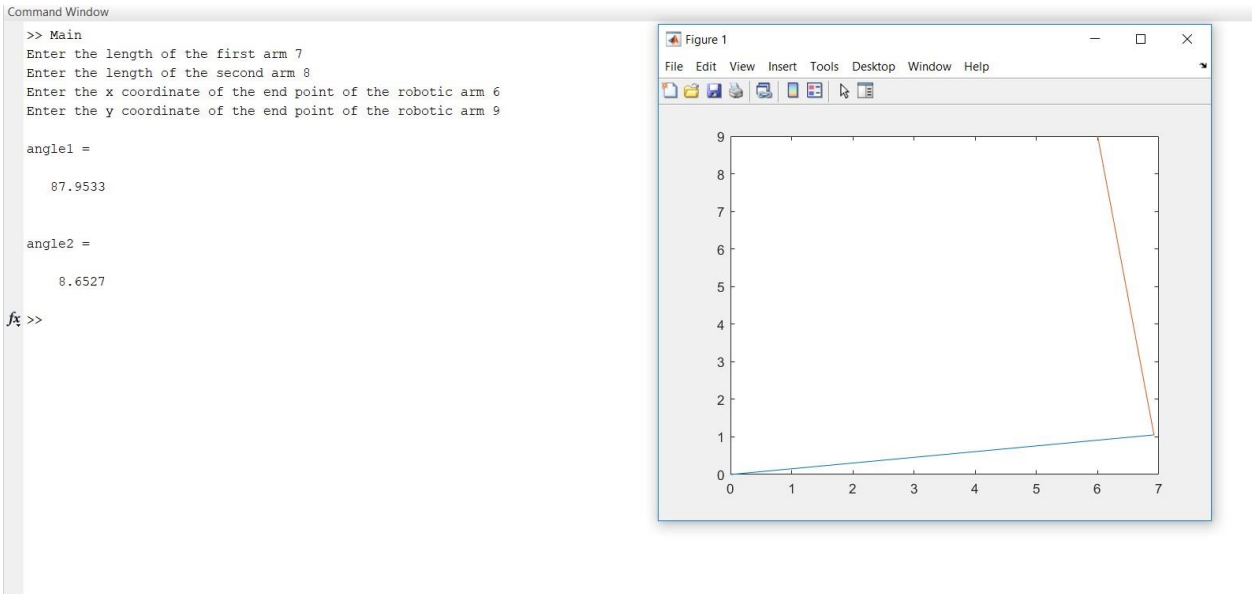
```
% to find final position link
link1=array1*link11;
link2=array1*array2*link22; end
 function[figure]=Graph(link1,link2)
%displaying the graph of the link
figure=plot(link1(1,:),link1(2,:),link2(1,:),link2(2,:));
end
```

Test Cases

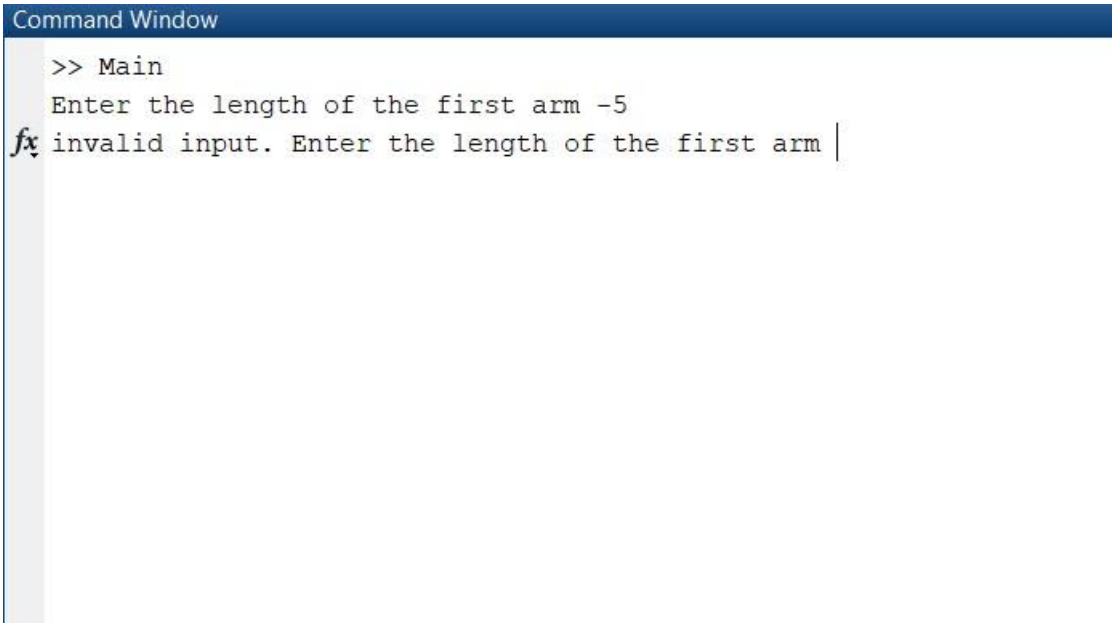
Test Case 1



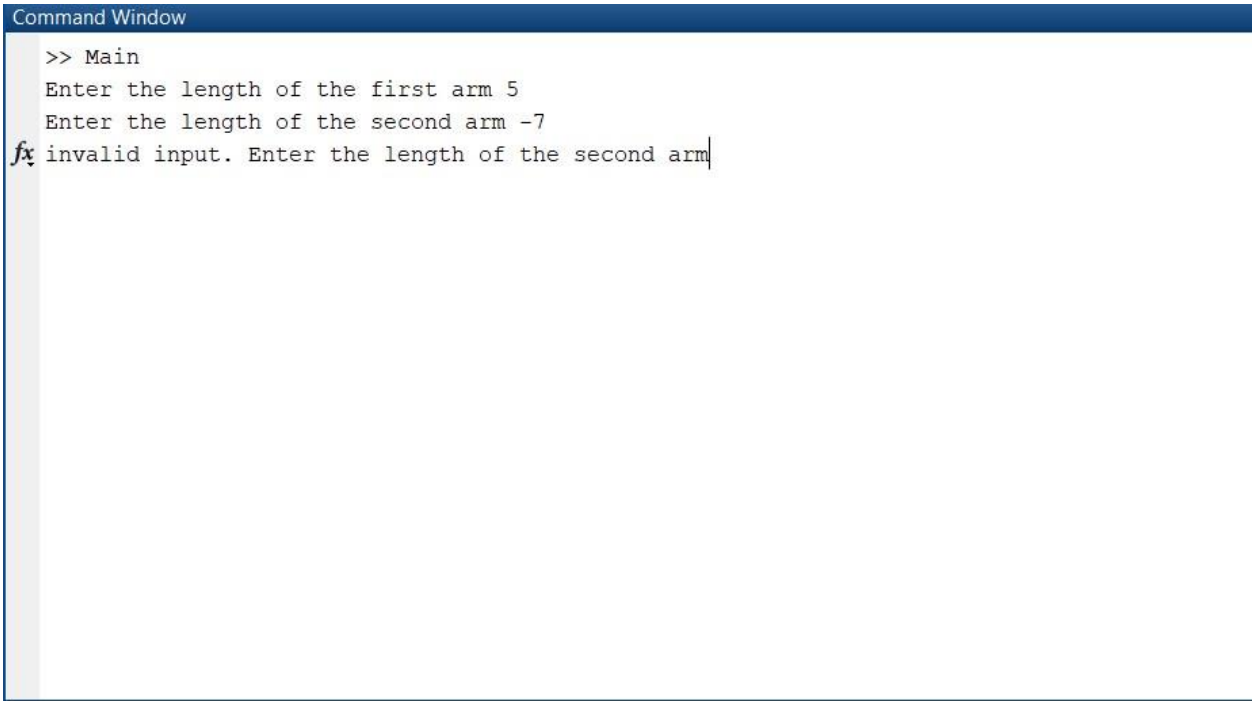
Test Case 2



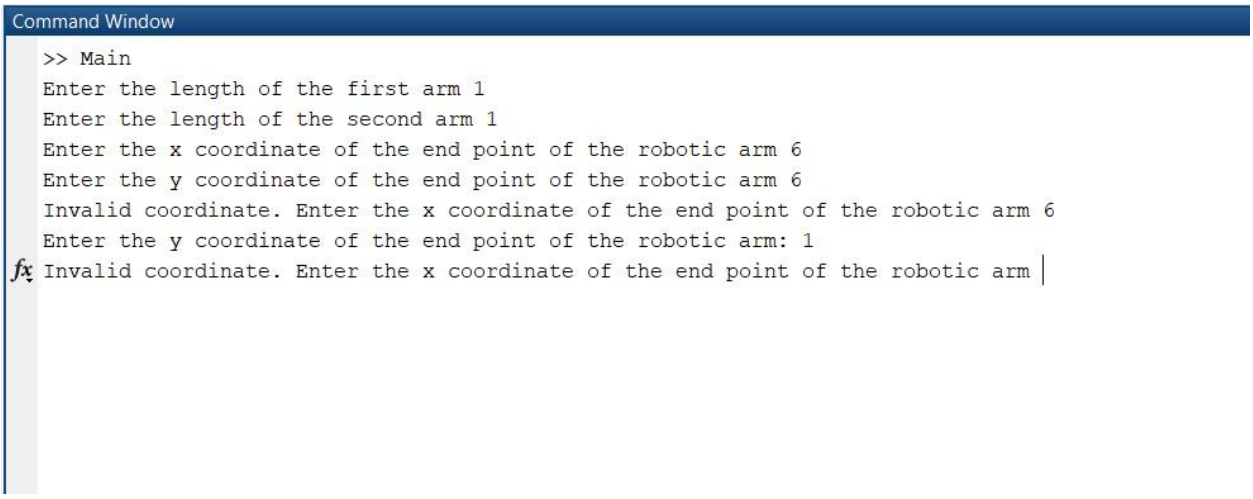
Test Case 3



Test Case 4



Test Case 5


Command Window
>> Main
Enter the length of the first arm 1
Enter the length of the second arm 1
Enter the x coordinate of the end point of the robotic arm 6
Enter the y coordinate of the end point of the robotic arm 6
Invalid coordinate. Enter the x coordinate of the end point of the robotic arm 6
Enter the y coordinate of the end point of the robotic arm: 1
Invalid coordinate. Enter the x coordinate of the end point of the robotic arm