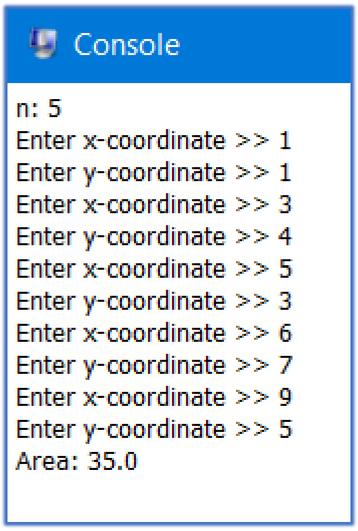
COL216

ASSIGNMENT-1

ABHISHEK KUMAR (2019CS10458)   ABHINAV SINGHAL (2019CS50768)

Problem Statement: Write a MIPS Assembly Program for obtaining the area under a curve formed by joining successive points by a straight line.

• **Major Design Considerations:**



```
Console
n: 5
Enter x-coordinate >> 1
Enter y-coordinate >> 1
Enter x-coordinate >> 3
Enter y-coordinate >> 4
Enter x-coordinate >> 5
Enter y-coordinate >> 3
Enter x-coordinate >> 6
Enter y-coordinate >> 7
Enter x-coordinate >> 9
Enter y-coordinate >> 5
Area: 35.0
```

I/O Interface for sample test case

A) Input/Output: The user interface is kept as simplistic as possible. First the number of points 'n' is entered by the user. Subsequently, all the points are entered one by one, each x and y coordinate on a separate line. Once the coordinates of all the points have been given by the user, the area (as a floating-point number) bound by the curve and X-axis is displayed on the screen.

B) Calculation of Area: Since we need to find area under the curve, we have taken all area to be positive and added even the area below the x-axis to our calculated area.

C) Implementation: After taking the integer input for 'n' and the first point's coordinates, we use a loop like a while loop and take one more point's coordinates per loop and calculate area under the curve between the previous point and the point just taken by using the appropriate math.

There are essentially two cases: - Y-coordinates of the two successive points have the-

- Same sign: add the modulus of area of the trapezium formed to our final area variable.
- Different sign: add the modulus of area of the two triangles formed (one above the x-axis and one below) to our final area variable.

Once we have taken all the input points and the areas have been successively added, we exit the loop and display the area which was stored in a 32-bit floating-point register. Our design considers integer coordinates for points which can be stored in a 32-bit register (i.e., up to $\pm (2^{31} - 1)$) and considers area to be stored in a 32-bit floating-point register (i.e., up to $\pm (2^{31} - 1)/100 = 21{,}474{,}836.46$).

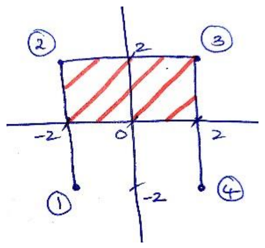
D) Raising Exceptions: We raise exceptions at appropriate locations:

- If n < 2, then we raise a bad input exception since it does not make sense to calculate area with less than two or a negative number of points.

- If the input points are not sorted according to the x coordinate, then we again raise a bad input exception since according to the assignment description, the input points had to be x-sorted.

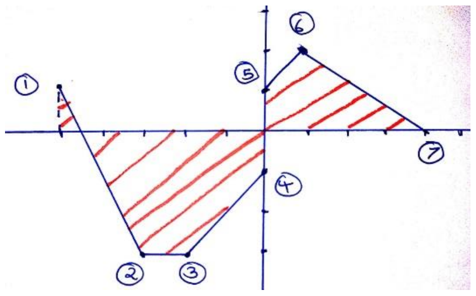
- **Test Cases:** We have rigorously tested our implementation using multiple test cases, a variety of them shown below. The design does not differentiate based on sign of coordinates of points. Hence, the points can be in any of the four quadrants as long as they are x-sorted and can fit into a 32-bit integer register.

  - Testcase – 1:
    - ⇨ N = 5
    - ⇨ (1,1) (3,4) (5,3) (6,7) (9,5) (Example given in assignment)
    - ⇨ Area: = 35.0
  - Testcase – 2:
    - ⇨ N = 4
    - ⇨ (-2, -2) (-2,2) (2,2) (2, -2)
    - ⇨ Area: = 8.0



Testcase - 2

  - Testcase – 3:
    - ⇨ N = 7
    - ⇨ (-5,1) (-3, -3) (-2, -3) (0, -1) (0,1) (1,2) (4,0)
    - ⇨ Area: = 14.0



Testcase -3

  - Testcase – 4:
    - ⇨ N = 2
    - ⇨ (0,2) (1, -1)
    - ⇨ Area: = 0.83
  - Testcase – 5:
    - ⇨ N = 11
    - ⇨ (-5, -5) (-4, -4) (-3, -3) (-2, -2) (-1, -1) (0,0) (1, -1) (2, -2) (3, -3) (4, -4) (5, -5)
    - ⇨ Area: = 25.0
  - Testcase – 6:
    - ⇨ N = 5
    - ⇨ (0,0) (1,0) (2,0) (3,0) (4,0)
    - ⇨ Area: = 0.0
  - Testcase – 7:
    - ⇨ N = 5
    - ⇨ (1,2) (5,6) (3,7) (9,10) (9,11)
    - ⇨ badInputException :: Given input points are not x-sorted -> Program Terminated

Apart from these many more have test cases have been tried and manual calculations have also been done to tally the answers.

-THANK YOU-