| | | |
|---|---|---|
| **Student 1:** | Pranay Gupta | 2019CS10383 |
| **Student 2:** | Divyanka Chaudhari | 2019CS50429 |



## Working of the MIPS code

The console first prints the prompt for the user to enter the number of points. The total no. of points inputted are stored in the register $\$s0$. We store a counter (for the loop) that will help us to know when to stop calculating in the register $\$s1$.

Next, we check if the number of points entered is $1$ or less than $1$. In this case, $0$ is outputted directly (we directly go to the single label) without asking the user to input the number because area under a single point would be 0.

Now, we start taking the input co-ordinates.
Co-ordinates must be inputted in the following manner: $x_1, y_1, x_2, y_2, ..., x_n, y_n$. We are storing the input co-ordinates in 32 bit integer registers, so it necessary they don't exceed $-2,147,483,648$ to $2,147,483,647$. The user should take care of this.

We take the input of the initial (or first) point as $(x, y)$ co-ordinates in the register $\$t8$ and $\$t9$. Next we go into the calculation loop where the final calculation would take place. The coordinates of the directly next point are stored in $\$t0$ and $\$t1$ registers. Then we find out the value of $\frac{(x_2-x_1)}{2}$ and store it in a double register $\$f2$. This value is stored in a double so as to allow a greater range of values to be calculated.

Meanwhile, we check if the y co-ordinates are of the same sign by multiplying their signs. If it is positive then both are on the same side of the x-axis and it goes to the positive label else if it is negative, then the points are on opposite sides of the x-axis and it goes to the negative label. We consider area below x-axis to be positive as well.

**Negative label:** The formula for area in this case is $0.5 * (x_2 - x_1) * \frac{(y_1^2+y_2^2)}{(|y_1|+|y_2|)}$. In the negative label, we calculate the value of $\frac{(y_1^2+y_2^2)}{(|y_1|+|y_2|)}$ and store it in the register $\$f4$ as a double. We make sure not to chance the values in t1 as that will be used during the calculation of area between the next $2$ points. We then directly jump to the extras label so that it does not go to the positive label.

**Positive label:** The formula for area this case is $0.5 * (x_2 - x_1) * (|y_1| + |y_2|)$. In the positive label, we calculate the value of $(|y_1| + |y_2|)$ and store it in the register $\$f4$ as a double.

**Extra label:** Then we go into the extras label in which we multiply the value stored in $\$f2$ and $\$f4$ and store it in$\$f2$. This gives us the (temporary) area under the graph for these $2$ points. We then add this area to the total area which has been stored in the $\$f0$ label and we increment the counter. Finally we move the values of $(x, y)$ coordinate from this loop into the registers $\$t8$, $\$t9$ which will be used as the first point during the next iteration. We keep the loop on and keep adding temporary our area to final area.

Now, we check if the counter is equal to the number of the points and exit accordingly. Finally, we print the double value stored in $\$f0$ which represents the total area.

## Testing

It is important to test exhaustively with all possible edge cases to determine mistakes. Here, we **do not** account for overflow occurring due input values entered beyond the integer range. Hence, our test cases are contained within this range. We also, as mentioned, assume that the points are sorted in ascending order of their x co-ordinates. In actual console, we input values by clicking enter unlike commas being used as delimiter to demonstrate in this document. Here, the first value of input is the number of points.

| Test case description | Input | Expected Output |
|---|---|---|
| Checking number of points to be less than or equal to 1 | 0 | 0 |
| | 1 | 0 |
| | -1 | 0 |
| | -2000000000 | 0 |
| Checking y co-ordinates to be 0 | 2, -1, 0, 2, 1 | 1.5 |
| | 2, -2, 0, 10, 0 | 0 |
| | 4, 2, 0, 4, -5, 7, 1, 8, 0 | 12 |
| Checking same values of x co-ordinates | 3, 2, 1, 2, 7, 2, 10000 | 0 |
| | 4, 0, 1, 0, 2, 0, 100000, 0, 20000000 | 0 |
| | 3, 1000, 1000, 1000, 1001, 1000, 1002 | 0 |
| Checking all points to be the same (including zero) | 2, 100, 90, 100, 90 | 0 |
| | 10, 1000000000, 0, 1000000000, 0 ... | 0 |
| | 100, 0, 0, 0 .... | 0 |

We did a partially exhaustive testing using a tester function (tester.cpp in appendix) we generated using CPP. It's hard to cover the range of the number of points, but we did cover co-ordinate values in this. We did this because we are familiar with testing regular higher level functions and do not know what other problems we could encounter other than the above ones while writing code in MIPS. We did not encounter problems here.

## Appendix

### Test case generator

```
#include <iostream>
#include <cstdlib>
#include <algorithm>
using namespace std;

int main () {
int randomNumberOfPoints;
srand(time(0)); randomNumberOfPoints = rand()%100;
cout<< "The number of points we're going to have are (only between 0 to 100):"
    <<randomNumberOfPoints<<endl;
//Assuming max value of RAND_max to be 2^32
srand(time(0)+1);
cout<<"Generating x co-ordinates"<<endl;
int xPoints[randomNumberOfPoints];
for(int i=0; i<randomNumberOfPoints;i++){ xPoints[i] = rand() - 65536;}
sort(xPoints, xPoints+randomNumberOfPoints);

srand(time(0)+2);
cout<<"Generating y co-ordinates"<< endl;
int yPoints[randomNumberOfPoints];
for(int j=0; j<randomNumberOfPoints; j++) {
  yPoints[j]= rand()-65536;
}
cout<<"The co-ordinates in the right order are:"<<endl;
for(int k=0; k<randomNumberOfPoints; k++){ cout<<xPoints[k]<<endl; cout<<
    yPoints[k]<<endl;}

cout<<"Calculating area..."<<endl; double area = 0.0;

for(int i=0; i<randomNumberOfPoints-1; i++){
  if((yPoints[i]/abs(yPoints[i])*(yPoints[i+1]/abs(yPoints[i+1]))) >= 0) {
    area = area + 0.5*(xPoints[i+1] - xPoints[i])*(yPoints[i] + yPoints[i+1]);
  }
  else {
    area = area + 0.5*(xPoints[i+1] - xPoints[i])*(((yPoints[i]*yPoints[i]) +
        (yPoints[i+1]*yPoints[i+1])/(yPoints[i]+yPoints[i+1])));
  }
}
cout<<"Expected area output:" << area;
}
```

**Sample output**

The number of points we're going to have are:23
Generating x co-ordinates
Generating y co-ordinates
The co-ordinates in the right order are:
25627880
54311153
189244683
1675509492
259560155
272924614
326524464
352848913
485662722
1250187230
701713115
41913505
751004163
699185643
862955460
1511002984
1011413345
1912558150
1102755505
1210248648
1110169007
1340117023
1262080429
1045626530
1304698605
577270941
1442301614
130646153
1633970454
1607067240
1720822652
1197049558
1779213380
324115469
1868744972
582494444
1881268357
1493882373
1961880586
140837372

1966952011
1701846173
2102745081
1888448077
2118085723
192533054
Calculating area...
Expected area output:4.51723e+17