

Programming Assignment 3

Hackerrank username: w_wangondu*
*submitted by classmate Fiona Ng'ang'a

Summary of your results (from Fiona's hackerearth):

The screenshot shows the 'Missing Soldiers' problem on Hackerrank. The problem description states: 'An infinite army of ants is marching on an infinite 2-D plane. Since ants are disciplined, here's how they march: each ant chooses exactly one x coordinate and moves along it in positive y direction, starting from (x, 0). There exists exactly one ant for each x coordinate on that plane and hence there are infinite ants!'. It also mentions N horizontal barriers. The task is to find the total number of ants that will be ever blocked.

The submission result table shows a 'Runtime error - NZEC' for the first submission. The table below summarizes the submission results:

Score	Time (sec)	Memory (KiB)	Language
0.0	0.45318	3204	Python 3.8

Input	Result	Time (sec)	Memory (KiB)	Score	Your Output	Correct Output	Log	Diff
Input #1	Runtime error	0.025862	2948	0				
Input #2	Runtime error	0.082664	3204	0				
Input #3	Runtime error	0.035436	3204	0				

N/B - (the submission 1 which says 2 minutes ago and has an orange error symbol is mine, the other one is Fiona's submission for her assignment)

The screenshot shows the 'My Submissions' table for the 'Missing Soldiers' problem. The table lists two submissions: one with a runtime error (orange icon) and one that was accepted (green checkmark).

#	Problem	Result	Time (Sec)	Memory (kb)	Language	Detail	Date
1	Missing Soldiers		0.45318	3204	Python 3.8	View	2 minutes ago
2	Missing Soldiers		2.09613	23012	Python 3.8	View	4 days ago

Below the table, there is a section for 'Best Submissions' with a language dropdown set to 'Bash (GNU bash, version 4.3.11)'. A message states: 'There is no solution for this language'.

Code (from my pycharm):

```
inputA = ["2\n", "1 1 4\n", "7 3 5\n"] # simulating hackerrank input()

lines = inputA

number_of_boundaries = lines[0] # can be used in a loop

start_x_coordinates = []
end_x_coordinates = []
y_coordinate_end_barriers = []

for i in lines:
    i.strip() # removes the newline character
    if i != lines[0]:
        a, b, c = i.split()
        x_boundary_start = int(a)
        x_boundary_stop = int(a) + int(c)
        start_x_coordinates.append(x_boundary_start)
        end_x_coordinates.append(x_boundary_stop)
        y_coordinate_end_barriers.append(int(b))

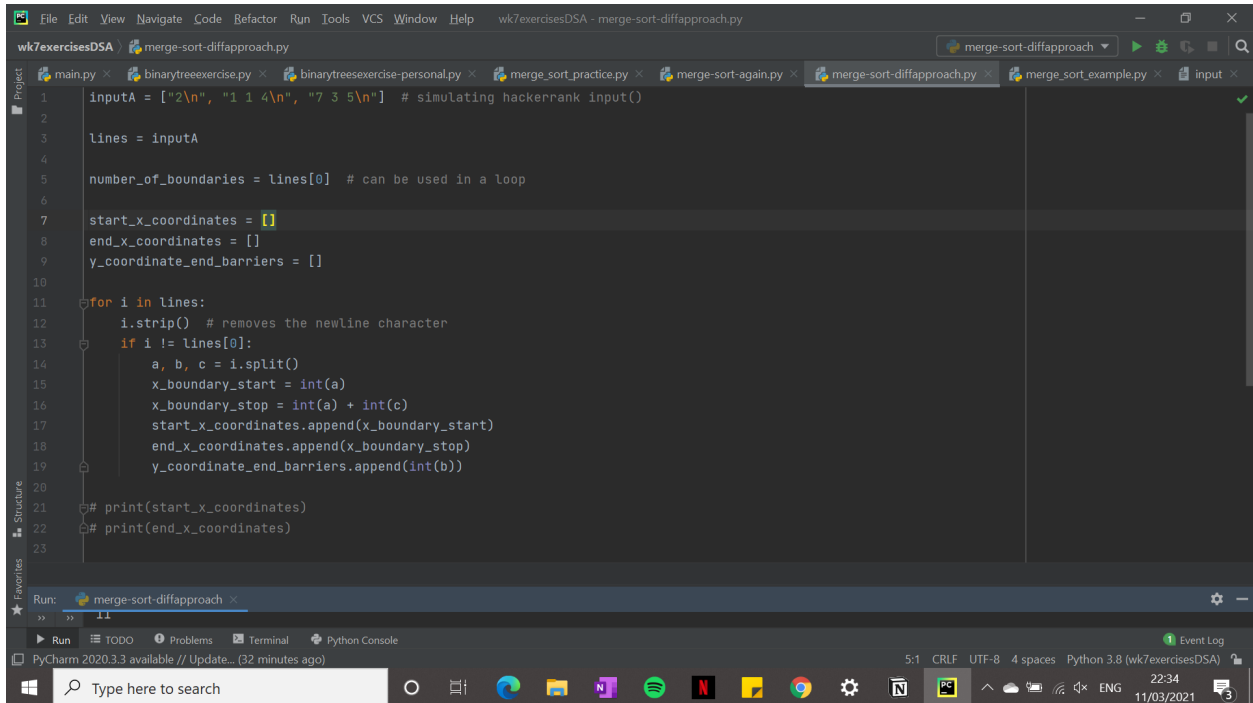
# print(start_x_coordinates)
# print(end_x_coordinates)

blocked_ants = []

for i in range(len(start_x_coordinates)):
    number_of_ants = abs(start_x_coordinates[i] - end_x_coordinates[i]) + 1
    blocked_ants.append(number_of_ants)

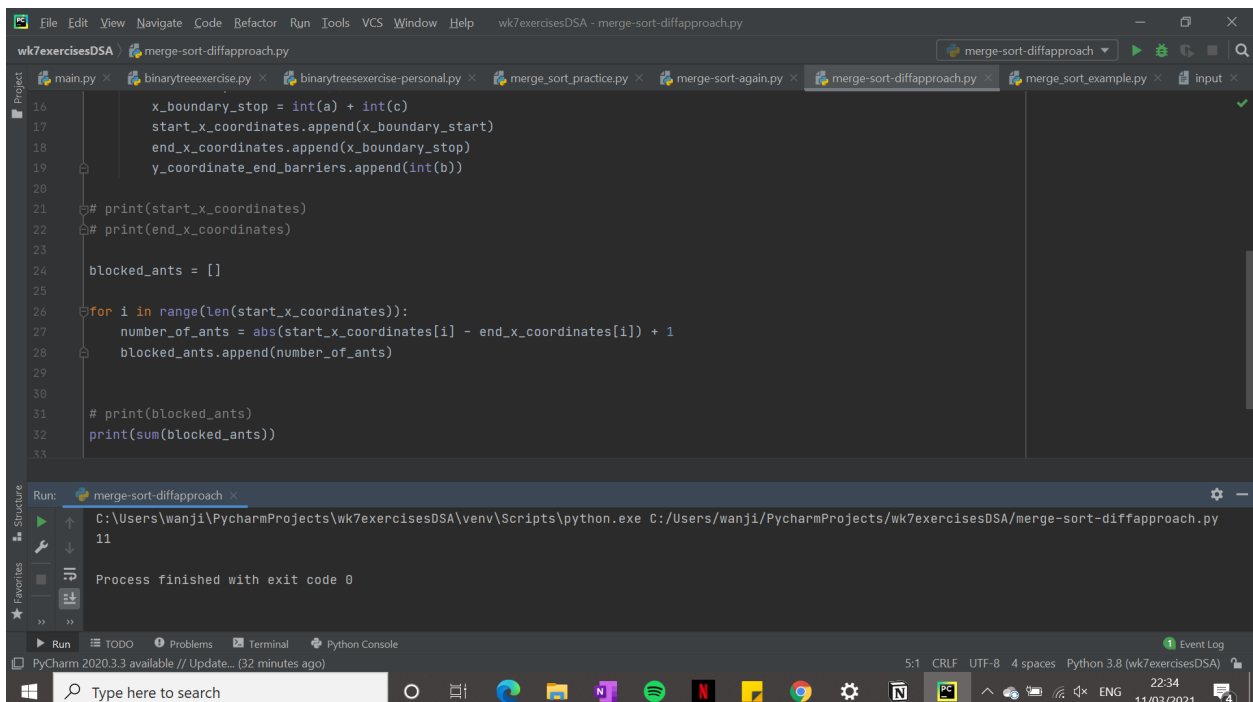
# print(blocked_ants)
print(sum(blocked_ants))
```

IDE Test Results:



This screenshot shows the first 23 lines of the `merge-sort-diffapproach.py` file in the PyCharm IDE. The code defines the input, initializes coordinate lists, and processes the input lines to populate these lists. The Run window at the bottom shows the program executed successfully with exit code 0.

```
1 inputA = ["2\n", "1 1 4\n", "7 3 5\n"] # simulating hackerrank input()
2
3 lines = inputA
4
5 number_of_boundaries = lines[0] # can be used in a loop
6
7 start_x_coordinates = []
8 end_x_coordinates = []
9 y_coordinate_end_barriers = []
10
11 for i in lines:
12     i.strip() # removes the newline character
13     if i != lines[0]:
14         a, b, c = i.split()
15         x_boundary_start = int(a)
16         x_boundary_stop = int(a) + int(c)
17         start_x_coordinates.append(x_boundary_start)
18         end_x_coordinates.append(x_boundary_stop)
19         y_coordinate_end_barriers.append(int(b))
20
21 # print(start_x_coordinates)
22 # print(end_x_coordinates)
23
```



This screenshot shows the continuation of the `merge-sort-diffapproach.py` code from line 16 to 33. It calculates the number of ants for each boundary and prints the total sum. The Run window at the bottom shows the program executed successfully with exit code 0.

```
16     x_boundary_stop = int(a) + int(c)
17     start_x_coordinates.append(x_boundary_start)
18     end_x_coordinates.append(x_boundary_stop)
19     y_coordinate_end_barriers.append(int(b))
20
21 # print(start_x_coordinates)
22 # print(end_x_coordinates)
23
24 blocked_ants = []
25
26 for i in range(len(start_x_coordinates)):
27     number_of_ants = abs(start_x_coordinates[i] - end_x_coordinates[i]) + 1
28     blocked_ants.append(number_of_ants)
29
30 # print(blocked_ants)
31 print(sum(blocked_ants))
32
33
```

Note: On the IDE, the code worked. I would appreciate it if you would test the code in your own IDE, or at least tell me why it gives the right answer on the IDE but not on Hackerearth.

