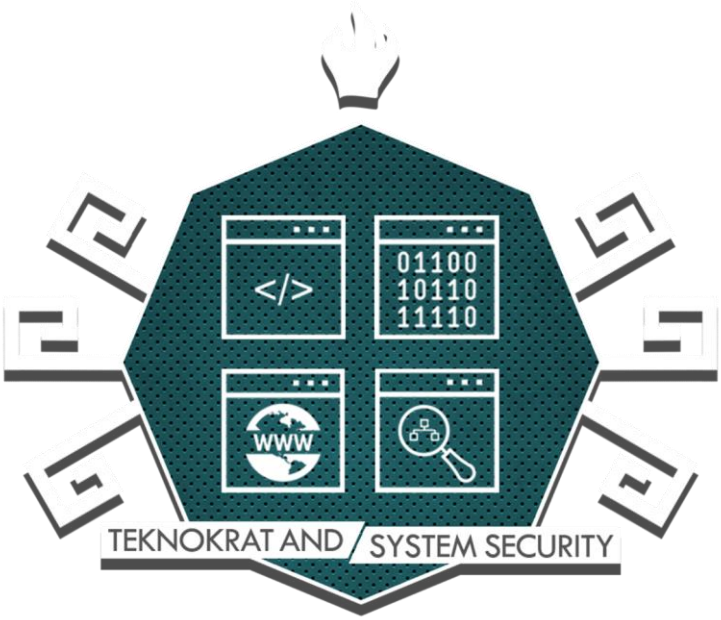
CTF Writeups By

TENESYS

"Facebook Hacker Cup 2017"

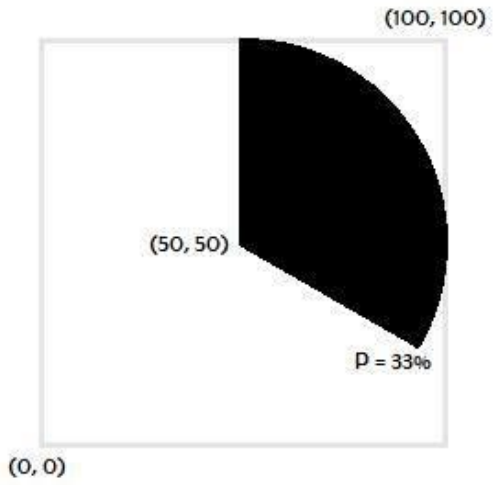


> Pada Facebook Hacker Cup 2017 Qualification Round, terdapat 3 problems, yaitu Progress Pie, Lazy Loading dan Fighting the Zombie. Setelah kita mendownload soal/input dari problem, kita hanya diberi waktu 6 menit untuk mensubmit hasil output dan source code dari script/program yang kita gunakan untuk memproses input tadi. Validasi benar/salahnya output akan dilakukan setelah babak kualifikasi ini berakhir, kami hanya bisa menyelesaikan 2 problem dan ternyata yang benar hanya 1.

Syarat untuk lolos ke round selanjutnya yaitu round 1 adalah menjawab minimal satu problem dengan benar.

Progress Pie
25 points

Some progress bars fill you with anticipation. Some are finished before you know it and make you wonder why there was a progress bar at all. Some progress bars progress at a pleasant, steady rate. Some are chaotic, lurching forward and then pausing for long periods. Some seem to slow down as they go, never quite reaching 100%.
Some progress bars are in fact not bars at all, but circles.
On your screen is a progress pie, a sort of progress bar that shows its progress as a sector of a circle. Envision your screen as a square on the plane with its bottom-left corner at (0, 0), and its upper-right corner at (100, 100). Every point on the screen is either white or black. Initially, the progress is 0%, and all points on the screen are white. When the progress percentage, **P**, is greater than 0%, a sector of angle (**P**% * 360) degrees is colored black, anchored by the line segment from the center of the square to the center of the top side, and proceeding clockwise.



While you wait for the progress pie to fill in, you find yourself thinking about whether certain points would be white or black at different amounts of progress.

Input

Input begins with an integer **T**, the number of points you're curious about. For each point, there is a line containing three space-separated integers, **P**, the amount of progress as a percentage, and **X** and **Y**, the coordinates of the point.

Output

For the **i**th point, print a line containing "Case #**i**: " followed by the color of the point, either "black" or "white".

Constraints

$1 \le T \le 1{,}000$
$0 \le P, X, Y \le 100$
Whenever a point (**X**, **Y**) is queried, it's guaranteed that all points within a distance of 10^{-6} of (**X**, **Y**) are the same color as (**X**, **Y**).

Example input

```
5
0 55 55
12 55 55
13 55 55
99 99 99
87 20 40
```

Example output

```
Case #1: white
Case #2: white
Case #3: black
Case #4: white
Case #5: black
```

Explanation of Sample

In the first case all of the points are white, so the point at (55, 55) is of course white.
In the second case, (55, 55) is close to the filled-in sector of the circle, but it's still white.

In the third case, the filled-in sector of the circle now covers (55, 55), coloring it black.

Input

```
1000
8 13 21
77 89 75
20 56 46
91 10 57
51 36 99
67 95 83
35 97 48
37 3 58
42 4 54
7 95 21
```

Catatan: Sebenarnya ada 1000 case pada file soal, untuk menghemat tempat, hanya dicontohkan 10 case saja.

Solution

```
from PIL import Image, ImageDraw
import os, sys
import Image

def pie(inp, inpx, inpy):
      N=8
      input = inp
      inputx = inpx
      y = 101
      inputy = y-inpy


      im=Image.new('RGB',(101*N,101*N),"#FFF")
      draw = ImageDraw.Draw(im,im.mode)
      awal = -90

      angle = (input/100.0)*360.0
      akhir = -90+int(angle)
      draw.pieslice((0,0,101*N,101*N),awal,akhir,fill="black")
      del draw
      im.save("file.png")

      warna = ""
      im = Image.open("file.png")
      rgb_im = im.convert('RGB')
      r, g, b = rgb_im.getpixel((inputx*N, inputy*N))
      if(r==0):
            warna="black"
      else:
            warna="white"
      return warna

filename = sys.argv[-1]
with open(filename) as f:
      lines = f.readlines()
num = lines[0].strip("\n")
```

```
for i in range(int(num)):
      angk = lines[i+1].strip("\n").split(" ")
      hasil=pie(int(angk[0]),int(angk[1]),int(angk[2]))
      print "Case #"+str(i+1)+": "+hasil
```

Result: Setelah proses validasi dilakukan, ternyata output yang kami kirim salah.

Lazy Loading
30 points

Wilson works for a moving company. His primary duty is to load household items into a moving truck. Wilson has a bag that he uses to move these items. He puts a bunch of items in the bag, moves them to the truck, and then drops the items off.

Wilson has a bit of a reputation as a lazy worker. Julie is Wilson's supervisor, and she's keen to make sure that he doesn't slack off. She wants Wilson to carry at least 50 pounds of items in his bag every time he goes to the truck.

Luckily for Wilson, his bag is opaque. When he carries a bagful of items, Julie can tell how many items are in the bag (based on the height of the stack in the bag), and she can tell the weight of the top item. She can't, however, tell how much the other items in the bag weigh. She assumes that every item in the bag weighs at least as much as this top item, because surely Wilson, as lazy as he is, would at least not be so dense as to put heavier items on top of lighter ones. Alas, Julie is woefully ignorant of the extent of Wilson's lack of dedication to his duty, and this assumption is frequently incorrect.

Today there are **N** items to be moved, and Wilson, paid by the hour as he is, wants to maximize the number of trips he makes to move all of them to the truck. What is the maximum number of trips Wilson can make without getting berated by Julie?

Note that Julie is not aware of what items are to be moved today, and she is not keeping track of what Wilson has already moved when she examines each bag of items. She simply assumes that each bagful contains a total weight of at least **k** * **w** where **k** is the number of items in the bag, and **w** is the weight of the top item.

Input

Input begins with an integer **T**, the number of days Wilson "works" at his job. For each day, there is first a line containing the integer **N**. Then there are **N** lines, the **i**th of which contains a single integer, the weight of the **i**th item, W_i.

Output

For the **i**th day, print a line containing "Case #**i**: " followed by the maximum number of trips Wilson can take that day.

Constraints

$1 \leq T \leq 500$

$1 \leq N \leq 100$

$1 \leq W_i \leq 100$

On every day, it is guaranteed that the total weight of all of the items is at least 50 pounds.

Example input

```
5
4
30
30
1
1
3
20
20
20
11
1
2
3
4
5
6
7
8
9
10
11
6
9
19
29
39
49
59
10
32
56
76
8
44
60
47
85
71
91
```

Example output

```
Case #1: 2
Case #2: 1
Case #3: 2
Case #4: 3
Case #5: 8
```

Explanation of Sample

In the first case, Wilson can make two trips by stacking a 30-pound item on top of a 1-pound item, making the bag appear to contain 60 pounds.
In the second case, Wilson needs to put all the items in the bag at once and can only make one trip.
In the third case, one possible solution is to put the items with odd weight in the bag for the first trip, and then the items with even weight in the bag for the second trip, making sure to put the heaviest item on top.

Input

```
500
80
54
73
88
56
51
88
5
38
```

Catatan: Sebenarnya ada 500 case (75673 baris) pada file soal, untuk menghemat tempat, input diatas hanya sebagai contoh saja.

Solution

```
import math, os, sys

def lazyloading(a):
      a = sorted(a, reverse=True)
      b=0
      i=0
      while(len(a)>0):
            if(a[i]>=50):
                  b+=1
                  del a[i]
            else:
                  nn = int(math.ceil(50.0/a[i]))
                  if(nn<=len(a)):
                        del a[i]
                        nn-=1
                        for j in range(nn):
                              del a[-1]
                        b+=1
                  else:
                        break
      return b

filename = sys.argv[-1]
with open(filename) as f:
      lines = f.readlines()
num = int(lines[0].strip("\n"))
first = 1
nam = 0
```

```
jl = 0
for i in range(num):
      first= first+nam
      jml = int(lines[first])
      temp = [0]*jml
      for j in range(jml):
            temp[j] = int(lines[first+j+1].strip("\n"))
      jl = lazyloading(temp)
      nam = int(lines[first].strip("\n"))+1
      print "Case #"+str(i+1)+": "+str(jl)
```

Result: Pada problem ini output yang kami kirimkan ternyata tepat. Jadi pada babak ini, hanya dapat menyelesaikan satu problem saja.