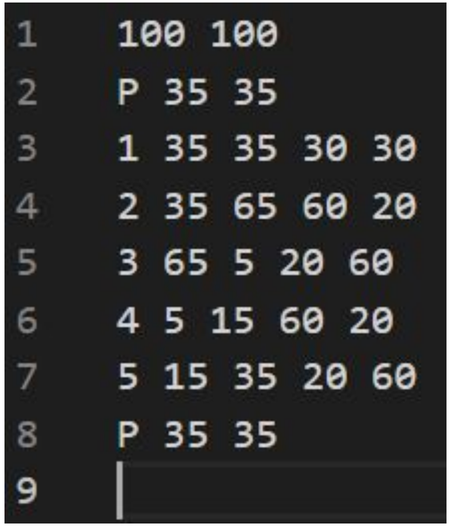
Lab1 Supplementary

Input

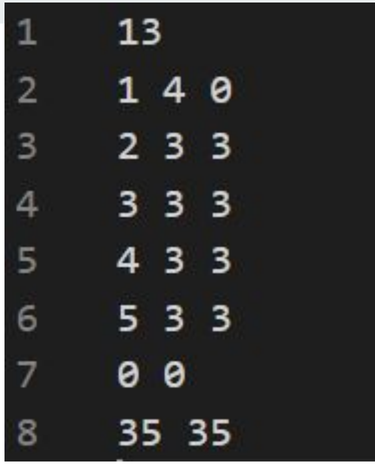
- First line is outline
  - [Width] [Height]
- Other lines has 2 type:
  - Point _Finding and Block_Tile_Creating
- Point _Finding:
  - P [X position] [Y position]
  - Point _Finding may appear on every line except the first line
- Block_Tile_Creating:
  - [Block_Tile_Index] [block’s x position] [block’s y position] [block’s width] [block’s height]
- Block_Tile_Index are positive integers and never repeat
  - Block_Tile_Index are not guaranteed to be consecutive and monotonicly increasing
  - ex: 1→2→100→4→5 is possible



```
100 100
P 35 35
1 35 35 30 30
2 35 65 60 20
3 65 5 20 60
4 5 15 60 20
5 15 35 20 60
P 35 35

```

Output

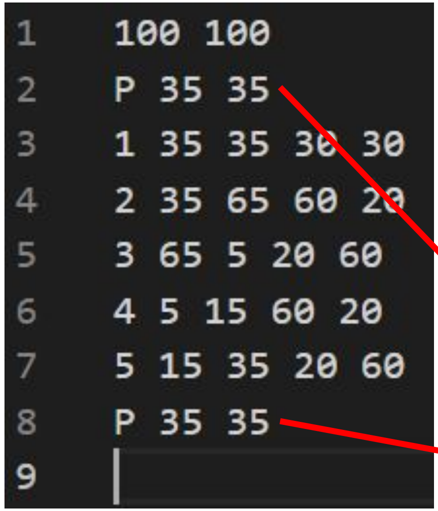


```
13
1 4 0
2 3 3
3 3 3
4 3 3
5 3 3
0 0
35 35
```

- Total block number (includind Block and Space tiles)
- Output every block tile **in block number ascending order**.
  - [Block_Tile_Index] [block neighbor's number] [space neighbor's number]
- Output belonging block's left bottom corner position of Point _Finding command continuously
  - Belonging block of Point _Finding is **depending on the input timing , not final situation!!**

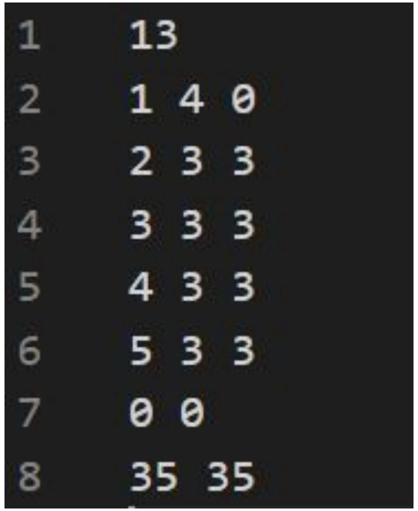
- **You could use linux command "diff" to check your output file's format is as same as TA.**

Example: case0



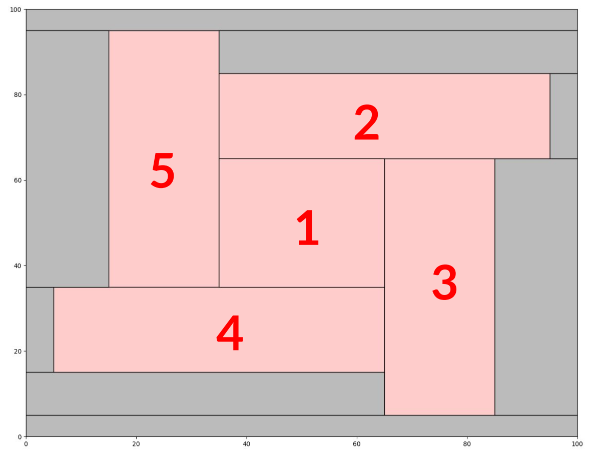
```
100 100
P 35 35
1 35 35 30 30
2 35 65 60 20
3 65 5 20 60
4 5 15 60 20
5 15 35 20 60
P 35 35
```

input file



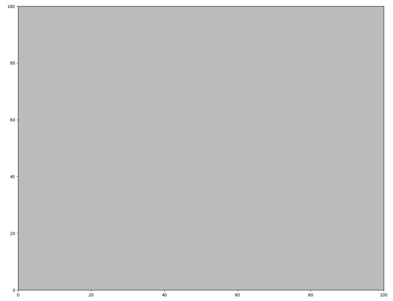
```
13
1 4 0
2 3 3
3 3 3
4 3 3
5 3 3
0 0
35 35
```

output file

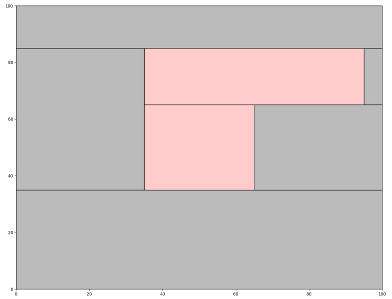


case0 step by step

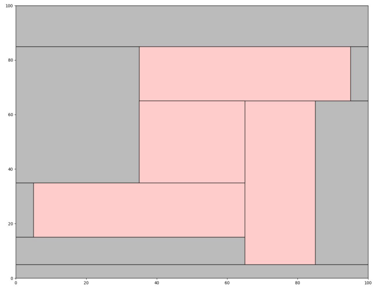
0.



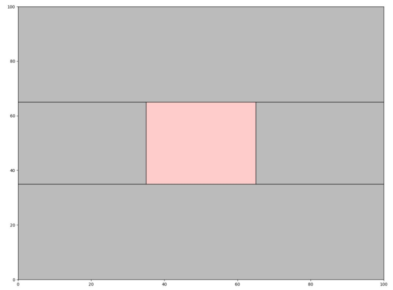
2.



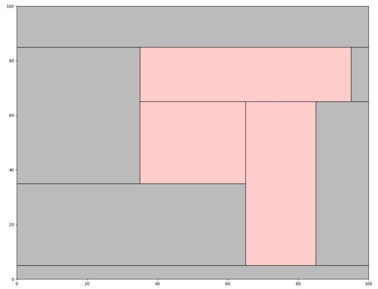
4.



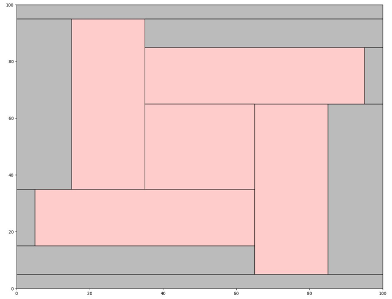
1.



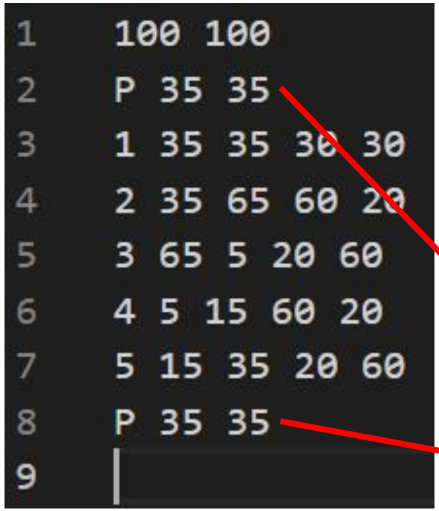
3.



5.

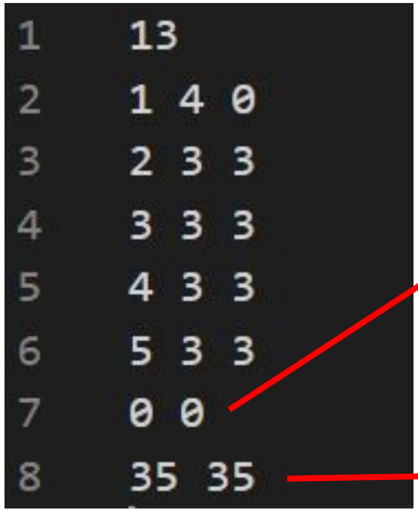


Example: case0



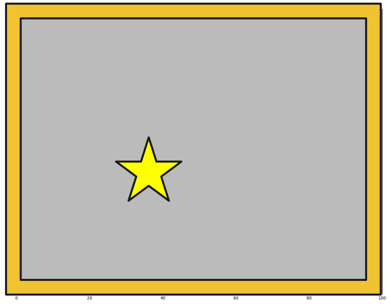
```
100 100
P 35 35
1 35 35 30 30
2 35 65 60 20
3 65 5 20 60
4 5 15 60 20
5 15 35 20 60
P 35 35

```

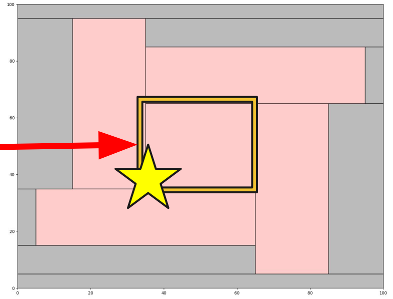


```
13
1 4 0
2 3 3
3 3 3
4 3 3
5 3 3
0 0
35 35
```

0.



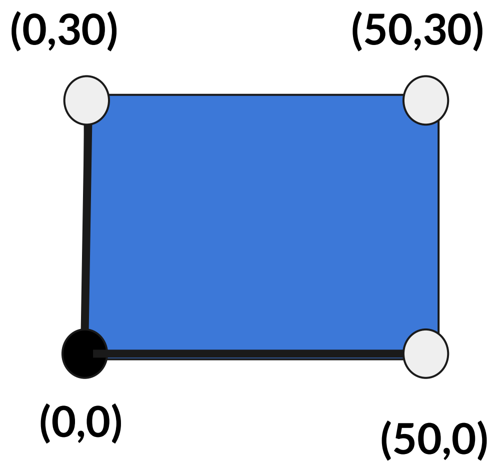
5.



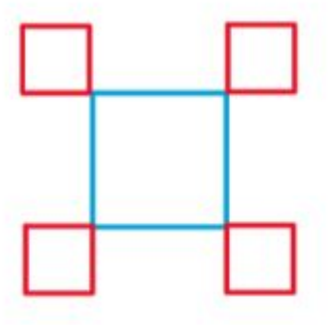
Boundary Condition Definition

- Right and top edge **not belong** to block

- Take the picture in the upper right corner as an example:
  - (0,0) , (0,25), (0,29)...... belong to this block
  - (0,30) , (50,10), (50,30) **not belong** to this block

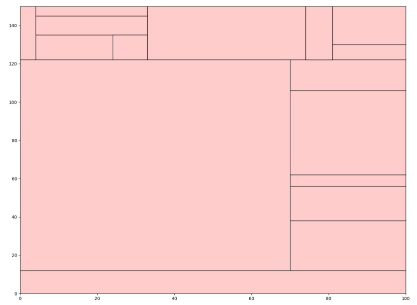


QA1



- Question
  - In this picture, Blue block and red blocks are neighbor?
- Answer
  - If 2 blocks are connected only at corners, they are not neighbors to each other.
  - None of block are neighbors in this picture.

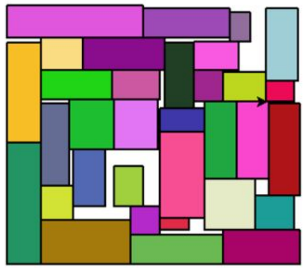
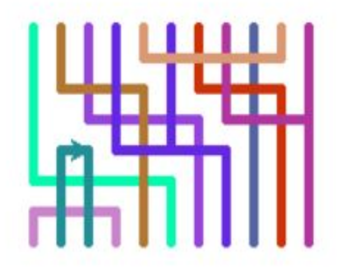
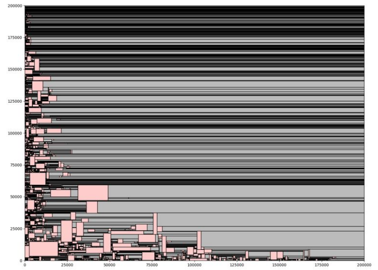
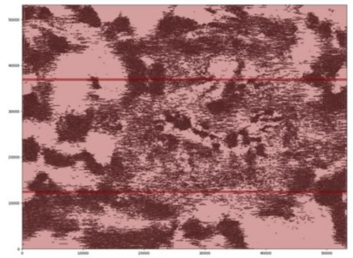
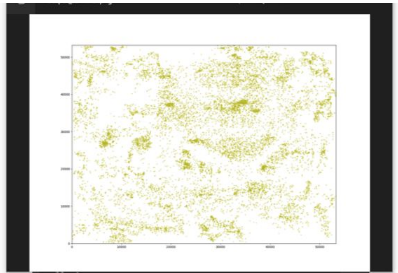
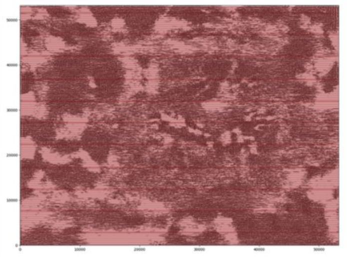
QA2



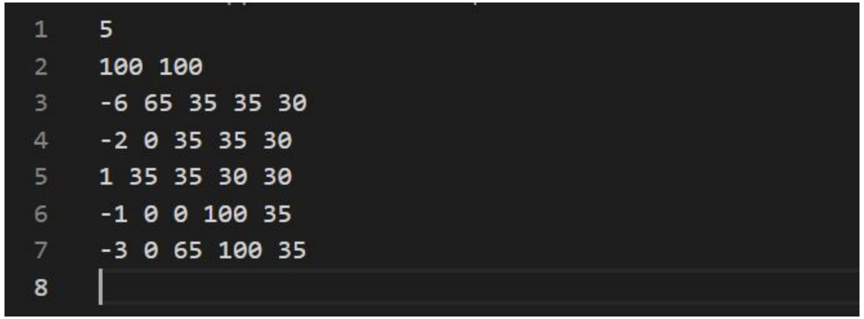
- Question
  - Block can stick together with other block or outline?
- Answer
  - Yes, block wouldn't overlap with each other or go out of outline, but sticking is allowing.
  - Picture in the upper right is a reasonable testcase.

Why Drawing?

- Visualization is important in back-end design, it helps to debug and confirm quality.
- Some example:

How to Drawing? (optional)



```
5
100 100
-6 65 35 35 30
-2 0 35 35 30
1 35 35 30 30
-1 0 0 100 35
-3 0 65 100 35
```

1. This section will not be graded.
2. You need to output another file which record every block's information like above picture, if you want to use drawing tool which provided by TA
   a. first line is the total block number
   b. second line is the outline of this case
   c. last lines are the block information
   d. [block_index] [block's x position] [block's y position] [block's width] [block's height]
   e. Positive block_index is for block tiles; Negative is for space tiles (At here, block_index are only used to distinguish between block(positive) and space(negative), you can also use 1 and -1)
3. Command: python3 draw_block_layout.py [LAYOUT_INFORMATION] [PICTURE_NAME]
   a. ex: python3 draw_block_layout.py layout0.txt layout0.png

4. You can also write other drawing programs by yourself