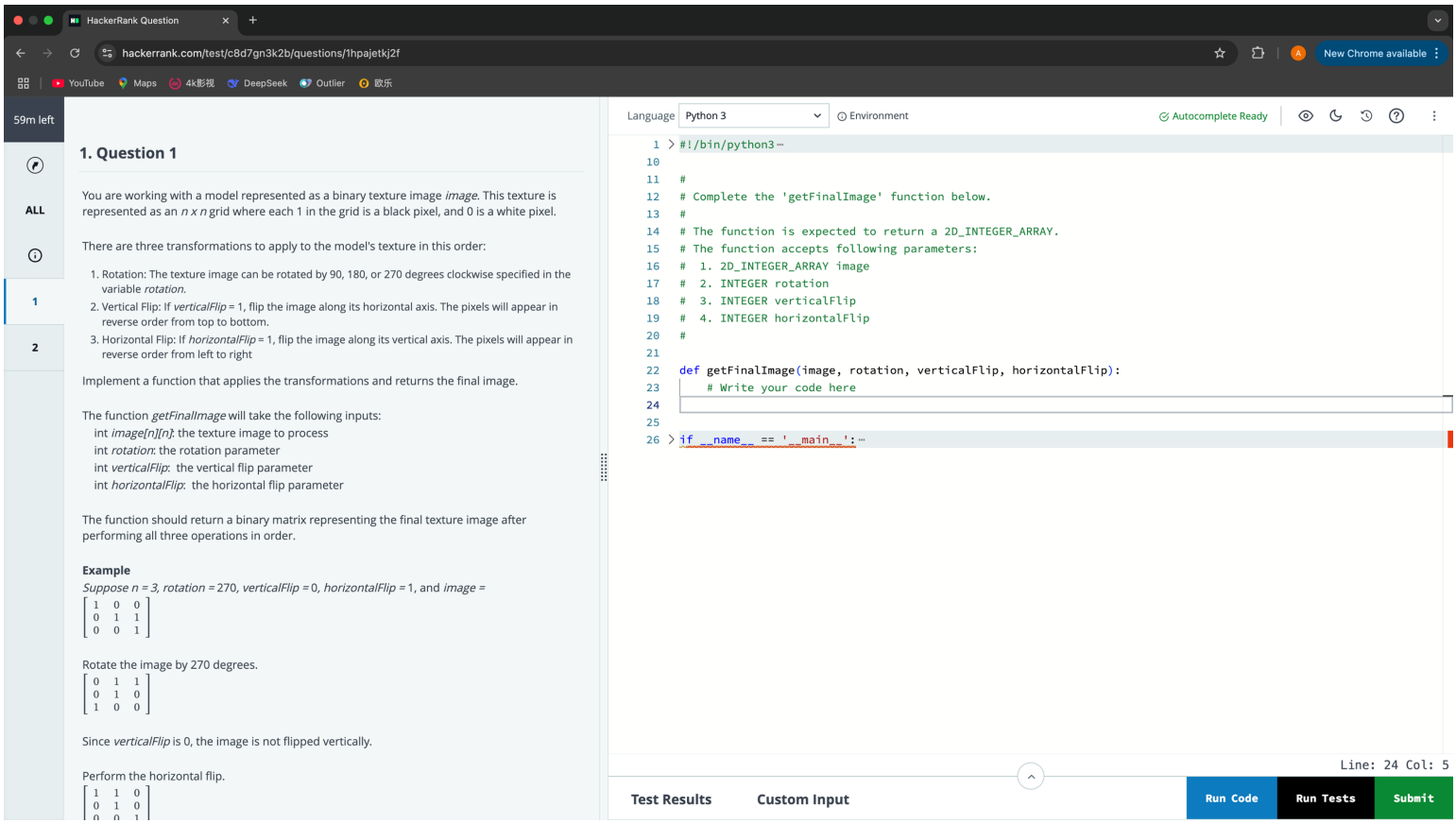
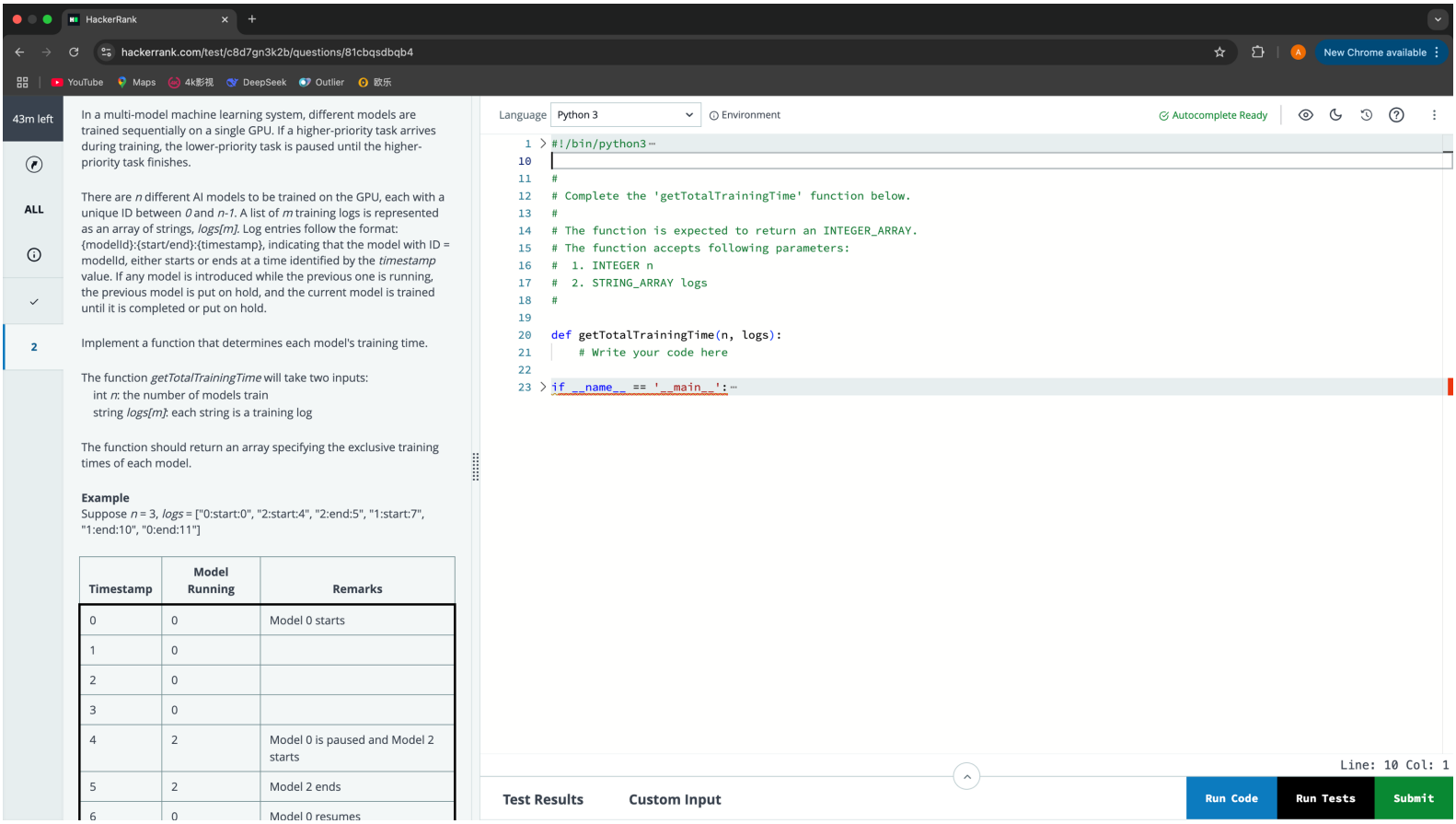


helper functions:

- Rotate:
  - 90 degree: m = [list(row)[::-1] for row in zip(*m)]
    - zip(*m) : transposing a matrix

```python
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
zip(*matrix)  →  (1, 4, 7), (2, 5, 8), (3, 6, 9)
```

    - 90 degree = transpose + reverse each row
    -180 degree = vertical flip + horizontal flip = 90 degree * 2
    - 270 degree = vertical flip + transpose = 90 degree * 3
    - else, no change
- Vertical
  - m[::-1]
- Horizontal
  -[row[::-1] for row in m]



use stack, obviously

steps:

- split log into segments: model, action, timestamp
- for each log, if start: pause current and push new one; else, pop current and resume prev if any
- track time deltas and add to corresponding model's counter

```
def getTotalTrainingTime(n, logs):
    stack = []
    res = []
    prev = 0

    for l in logs:
        seg = l.split(":")
        model, action, timestamp = int(seg[0]), seg[1], int(seg[2])

        if stack: #check if currently has a model running
            res[stack[-1]] += timestamp - prev # time btw the new model start and current running is available for current model to run
        if action == "start":
            stack.append(model)
        else: # a model finish running
            res[stack.pop()] += 1 #pop from stack and add to res, +=1 for account inclusive end
            timestamp += 1 # update timestamp to next logical time
        prev = timestamp
    return res
```