CSCI 561: Foundations of Artificial Intelligence
Homework #2: Airport Gate Assignment Problem
Due at March 8, 11:59 PM PT

Problem Description

LAX is the world's fifth-busiest airport. It is therefore impossible to manually coordinate all of the planes that are trying to land, make sure there is an open gate for each plane upon landing, and again coordinate the planes trying to take off again. LAX's air traffic controllers are asking for the USC CS department to design a program to help them do this job.

The problem starts with **N** airplanes flying over LAX, waiting to land, unload passengers, board new passengers, and take off again. Each plane has only limited fuel remaining, allowing it to stay in the air for at most **R** minutes. A plane must begin its landing procedure before its fuel runs out. The plane will arrive at a boarding gate **M** minutes after starting its landing procedure. It will then take **S** minutes to unload passengers, refuel, board new passengers, etc. before it is ready to take off again. If the plane spends more than **C** minutes at the gate, the waiting passengers will complain, which must be avoided at all cost. Once the plane leaves the gate, it will take **O** minutes for it to finally take off and for the inflight beverage service to begin.

Unfortunately, LAX does not have enough runways or boarding gates to allow all of the airplanes to land, board, and take off whenever they want. Due to the crowded airspace and the limited number of runways, there can be at most **L** planes landing and at most **T** planes taking off at the same time. Also, there are only **G** boarding gates, and each gate can accommodate only one plane at a time.

The problem facing the air traffic controllers is to assign each plane in the air a time when it should begin its landing procedure and a time when it should begin its takeoff procedure. Each plane must have enough fuel to stay in the air until its assigned landing time, a gate to stay at before its time to take off again, and an early enough take off time so that its waiting passengers do not complain.

Input Format

The input file name is "input.txt".
<AIRPORT INFORMATION>: contains 3 integers, **L G T**, separated by spaces. The first is the maximum number of planes that can be landing at the same time. The second is the number of

boarding gates in the airport, each capable of serving only a single plane. The third is the maximum number of planes that can be taking off at the same time.

<NUMBER OF PLANES>: contains one integer **N** specifying the number of planes. The next **N** lines will provide the information about each of the planes currently hovering over the airport.

<PLANE 1>: contains 5 positive integers, **R M S O C**, separated by spaces. **R** is the maximum number of minutes that this plane can keep hovering based on its remaining fuel. **M** is the number of minutes it takes the plane to reach its gate after initiating its landing. **S** is the number of minutes it takes the gate crew to serve the plane (unload passengers, refuel, board new passengers, etc.). **O** is the number of minutes it takes the plane to complete takeoff after leaving the gate. **C** is the maximum number of minutes that a plane can stay at the gate before passengers start complaining.

<PLANE 2>: Same format as Plane 1.

.....

<PLANE N>: Same format as Plane 1.

Sample Input file:

```
1 2 1
4
0 10 50 20 70
20 20 40 20 60
60 10 70 20 75
80 10 30 30 80
```

This input sample tells us: There can be at most 1 plane landing at a time, 2 planes on the ground at a time, and 1 plane taking off at a time. There are 4 planes in total. Plane 1 needs to begin landing right now. It needs 10 minutes to completely land on the ground and arrive at a gate. It must spend a minimum of 50 minutes at the gate before leaving. It needs 20 minutes to complete takeoff after leaving the gate. And its passengers will start complaining if it spends more than 70 minutes at the gate. The rest of the planes are similarly specified.

Output Format

Your program must output an output file "output.txt". You need to print out **N** lines containing the assigned landing and takeoff times (in minutes) for each plane. The problem starts at time 0.

<ARRANGEMENT PLANE 1>: Each line contains 2 integers, **A B**, separated by a space. **A** is the time (number of minutes from the start) that you have granted this plane to begin landing. **B** is the time (number of minutes from the start) that you have granted this plane to leave the gate

and this plane start to take off.
<ARRANGEMENT PLANE 2>: Same as the format for Plane 1.
.....
<ARRANGEMENT PLANE N>: Same as the format for Plane 1.

Sample Output file:

0 60
10 80
50 130
70 150

This is a solution to the problem specified in the Sample Input File. We assign plane 1 to land now (**A**=0), because plane 1 can no longer hover in the air (**R**=0). 10 minutes later (**M**), plane 1 arrives at its gate and begins its service. At 60 minutes, Plane 1 can leave the gate and take off, finishing its takeoff 20 minutes after that (**O**), or 80 minutes after the start. Plane 1 thus spends the minimum 50 minutes (**S**) at the gate, and it avoids the passenger complaints that would have begun after 70 minutes (**C**).

Because **L**=1, plane 2 needs to wait for plane 1 to finish landing first. At 10 minutes (**A**), plane 2 can start landing (we assume that plane 1 has already cleared the landing space when the clock hits 10). At 30 minutes (**M**=20), plane 2 reaches the gate and does not leave until 80 minutes from the start (**B**). Notice that plane 2 spends 50 minutes at the gate, which is more than its minimum service time (**S**=40), because it has to wait for plane 1 to finish taking off (**T**=1). But it still leaves early enough so that passengers do not complain (**C**=60).

Plane 3 needs to stay in the air a little longer because there is no available gate until 60 minutes after the start. It can therefore begin landing at 50 minutes and, 10 minutes later (**M**), it can use the gate that plane 1 has just left. While plane 4 can finish its service earlier than plane 3, we let plane 3 depart first, to avoid passenger complaints (**C**).

Homework Rules

1. Please follow the instructions carefully. Any deviations from the instructions may lead your grade to be zero for the assignment.
2. You must use Python 2.7 to implement your code. You are allowed to use standard libraries only. You have to implement any other functions or methods by yourself.

3. You need to create a file named “hw2cs561s2019.py”. The command to run your program would be as follows: (When you submit the homework on labs.vocareum.com, the following commands will be executed.)
   python hw2cs561s2019.py

4. Homework must be submitted through Vocareum. Homework submitted through any other channel will NOT be accepted. Please only upload your code to the “/work” directory. Don’t create any subfolders or upload any other files.

5. For your final submission, please do not print any logs on the console other than the required output.

6. Your program should handle each test case within one minute.

7. Homework submitted after the deadline will **NOT** be graded. It is recommended that you submit versions of your homework early to avoid any submission issues on Vocareum.