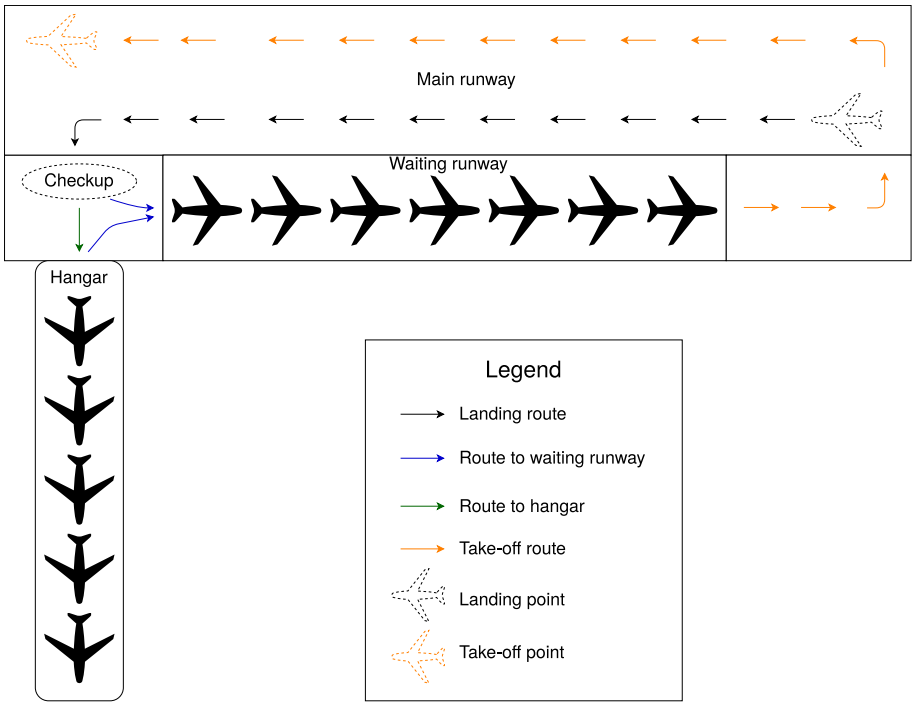
LAB 1: airport

As you can read in the news about Schiphol, running an airport can be difficult. In this assignment it is your job to manage a small airport. You need to decide in which order the planes landing at the airport will take off again. Be careful and do not cause accidents!

The Problem

Here is a simplified map of the airport:



Planes arrive and leave according to the following rule:

1. A plane lands on the **main runway** (black arrows).
2. The landed plane reaches the **checkup** point to see whether it requires any repairs.
3. If the plane does need repairs, then it moves into the **hangar**.
4. If the plane does not need any repairs, then it moves onto the **waiting runway**.
5. The waiting runway can hold up to seven planes. Whenever the *waiting runway is full*, the "Ready for takeoff!" signal is given and all planes on the waiting runway move to the main runway and depart.
6. The hangar can hold up to five planes. Whenever the *hangar is full*, first the waiting runway is cleared by letting any planes there depart, then all planes in the hangar move onto the waiting runway.
7. At the end of the day, after all planes have landed, all remaining planes leave the airport as follows: First the waiting runway is cleared, then all planes in the hangar move to the waiting runway and then again the waiting runway is cleared.

Your Task

Fortunately, the control tower already makes sure that only one plane will be on the main runway at any time. So you do not have to manage the main runway. But it is your job to manage the hangar and the waiting runway, and to decide in which order the planes should depart. Make sure you follow the rules above and the following additional assumptions.

- All planes have a unique name which is an integer.
- The hangar and waiting runway are never both full at the same time.
- The hangar has only one narrow entrance which is also its exit, and planes cannot move past each other within the hangar. For example, to ensure other planes can still enter, the first plane to arrive in the hangar will move as far into the hangar and away from the entrance as possible.
- Also on the waiting runway the planes are not allowed to overtake each other.

Input and Output format

The **input** starts with a number n (with $0 \leq n \leq 1000$) indicating the number of planes arriving at the airport. It is followed by n lines which each contain the unique integer name of the plane to land, followed by a space and a string "yes" or "no" saying whether the plane needs repairs.

The **output** gives the order in which the n planes may leave the airport, with one plane per line. Moreover, before each group of planes is taking off, the line "Ready for takeoff!" should be printed.

You can find two examples of the input and output at the end of this PDF.

Hints

- You can read in each plane using a command like `scanf("%d %s", &planeID, needsRepairs);`
- You will need *one stack and one queue* to solve this problem.
- Keep track of when the hangar or waiting runway are full with extra variables! (Because a stack and a queue do not provide a function to get the current number of items.)

Template

Please download `airport.zip` from Canvas and extract all files from it. Then write your own solution into `airport.c`. The library files for stacks and queues are already included and you must not modify them.

Testing

You can test your solution by running `make test`. This will check if your program works for the two input-output examples below.

For further testing, you can also submit your solution to the online "Exercise Testing" tool on Canvas. This will use the same two test cases, and three additional test cases. For the three additional tests the input and output are hidden, so you will only see whether your program passed or failed the test.

When grading your program I will use the same tests as the online tool and half of your grade (5 out of 10 points) will be given for passing the tests.

Input-Output Example 1

input	corresponding **output**
10 1 no 2 no 3 no 4 yes 5 yes 6 no 7 yes 8 yes 9 no 10 no	Ready for takeoff! 1 2 3 6 9 10 Ready for takeoff! 8 7 5 4

Input-Output Example 2

input	corresponding **output**
20 1 no 2 yes 3 no 4 no 5 yes 6 yes 7 no 8 yes 9 yes 10 yes 11 yes 12 no 13 no 14 no 15 no 16 yes 17 no 18 no 19 yes 20 yes	Ready for takeoff! 1 3 4 7 Ready for takeoff! 9 8 6 5 2 12 13 Ready for takeoff! 14 15 17 18 Ready for takeoff! 20 19 16 11 10

Report

Besides your program you should also submit a short report ($\leq$ 2 pages). You can find a LaTeX and an ODT template for this on Canvas between which you can choose. In any case, submit the report as a PDF file.

Submitting

To submit your work, please upload `airport.c` and `report.pdf`.