In the lecture this week, we apply bottom-up and top-down design to create a program to simulate the behavior of an airport runway. A succinct statement of the problem is:

Write a program to simulate take-off and landing queues at one runway of an airport. Allow planes to arrive in their queues at random times. The parameters describing each queue are the average plane arrival rate and the time each plane will occupy the runway.

and a sketch of the problem with the important model parameters is provided in Figure 1.

Figure 1. Sketch of an airport runway showing the take-off and landing queues and the model parameters describing the arrival rates of planes for each queue and the time each plane uses on the runway.

In the bottom-up design approach, we look for simple programming elements that we know will be needed in our programs. We started by listing general ideas about these functions. We then refined the list with function names, lists of input data required, and a one-line function description. We can convert this initial list of program capabilities into the start of our program code by creating the function definitions and first line of the doc-strings together with a code testing area. The listing for runway_draft.py demonstrates this first step in our program design.
Simulate take-off and landing on a single runway

Provides functions to simulate the take-off and landing queues for airplanes on a single runway. Planes arrive in their queues following a Poisson process. Planes waiting to land have priority over planes waiting to take off. Parameters of the simulation are provided by the user through the function parameters and include the number of planes arriving to land and take off per hour, the duration of a landing event, and the duration of a take-off event.

S. Socolofsky
ENGR 102
October 2018

""
import numpy as np
from scipy.stats import poisson

def arrival(queue, mu, delta_t=1):
    """ Simulate the arrival of planes for landing or take-off """

def waiting(queue, delta_t=1):
    """ Update the wait times for all planes in a queue """

def runway(queue, event_duration):
    """ Remove a plane from a queue and occupy the runway """

def tower(l_queue, t_queue, l_dur, t_dur, occupied):
    """ Let the tower direct take-off and landing """

def report(t, t_queue, l_queue, occupied):
    """ Report the status of each queue and the runway """

def update_stats(num, sum_wait, max_wait, event_wait):
    """ Update variables used to keep queue and runway statistics """

if __name__ == '__main__':
    landing = []
    takeoff = [4, 3, 2]
We also applied a top-down design approach to this program. Figure 2 shows the flow chart of program elements we developed through our top-down design process.

Figure 2. Top-down design flow chart for a program to simulate the queues at a single airport runway.

To create our runway simulation, we will write two Python script files: `runway_queue.py` contains all of our program functions, and `runway_sim.py` contains the code for the simulation. We include program code to test each function within `runway_queue.py`; this file becomes a module that holds the functions used in our runway simulation. The script `runway_sim.py` imports the functions in `runway_queue.py` and uses them to create the simulation.
Simulate take-off and landing on a single runway

Provides functions to simulate the take-off and landing queues for airplanes on a single runway. Planes arrive in their queues following a Poisson process. Planes waiting to land have priority over planes waiting to take off. Parameters of the simulation are provided by the user through the function parameters and include the number of planes arriving to land and take off per hour, the duration of a landing event, and the duration of a take-off event.

S. Socolofsky
ENGR 102
October 2018

""
import numpy as np
def get_sim_parameters():
    """
    Get the simulation parameters from the user
    Ask the user to input the values for the average arrival rates of planes and durations for take-off and landing. Uses console input to obtain parameter values.
    Returns
    -------
    t_dur : float
        Time required (min) for plane to take off
    l_dur : float
        Time required (min) for plane to land
    t_rate : float
        Average number of planes taking off (#/hr)
    l_rate : float
        Average number of plane landing (#/hr)
    sim_dur : float
        The desired simulation duration (hr)
    """
    print('
Enter the parameter values of the runway simulation: ')
t_dur = float(input('    Time required to take off (min): '))
l_dur = float(input('    Time required to land (min): '))
t_rate = float(input('    Average number of planes taking off (#/hr): '))
l_rate = float(input('    Average number of plane landing (#/hr): '))
sim_dur = float(input('    The desired simulation duration (hr): '))
```
' Average number of planes taking off per hour: '))
l_rate = float(input(' Average number of planes landing per hour: '))
sim_dur = float(input('Enter the number of hours to simulate: '))

return (t_dur, l_dur, t_rate, l_rate, sim_dur)

def arrival(queue, mu, delta_t=1):
    
    Simulate the arrival of planes for landing or take-off

    Determines the number of planes arriving for landing or take-off during
    the given time interval delta_t using a Poisson process with mean arrival
    rate mu. Adds the new plane(s) to the queue and initializes the wait time
    for each plane in the queue to zero.

    Parameters
    ----------
    queue : list
        List of planes waiting for take-off and landing. The value of each
        element in the list is the time (min) the plane has been in the list.
    mu : float
        Arrival rate of planes for landing or take-off (#/hour)
    delta_t : int, default=1
        Interval to simulate (min)

    Returns
    -------
    n_planes : int
        Number of planes arriving (#) for landing or take-off during the
        given time interval delta_t

    Notes
    -----  
    Since lists are mutable, no lists need to be returned from this function.
    
    # Simulate each minute over the interval delta_t as a separate event
    from scipy.stats import poisson
    r = poisson.rvs(float(mu) / 60., size=delta_t)
    
    # Add all new planes to the queue with their wait times
    for i in range(len(r)):
        if r[i] > 0:
            queue.append(delta_t - i - 1)
    
    # The total number of planes is the number of events (ones) in the array r
    return np.sum(r)

def waiting(queue, delta_t=1):
    
    Update the wait times for all planes in a queue
```
Add the time delta_t to all times stored in the queue of wait times

Parameters
----------
queue : list
    List of planes waiting for take-off and landing. The value of each
    element in the list is the time (min) the plane has been in the list.
delta_t : int, default=1
    Amount of time to add to each plane in the waits lists (min)

Notes
-----
Since lists are mutable, no lists need to be returned from this function.

```python
# Add delta_t to each element in waits
for i in range(len(queue)):
    queue[i] += delta_t
```

```
def runway(queue, event_duration):
    
    Remove a plane from a queue and occupy the runway

    Simulates a plane on the runway by removing the plane from its queue and
    wait-times lists. Returns the total time the plane was in the queue,
    including the time on the runway.

    Parameters
    ----------
    queue : list
        List of planes waiting for take-off and landing. The value of each
        element in the list is the time (min) the plane has been in the list.
    event_duration : int
        Time it takes (min) for planes to take off or land

    Returns
    -------
    queue_time : int
        Total time plane spent in the queue (min), including time for landing
        or take-off
    event_duration : int
        Time it takes (min) for planes to take off or land

    Notes
    -----  
    Since lists are mutable, no lists need to be returned from this function.
    ```
    # Remove the first plane in the queue and get its wait time
    queue_time = queue.pop(0)
    # Add the time for take-off or landing to the queue time
    queue_time += event_duration
    # Return the total wait time for this plane
```
return (queue_time, event_duration)
def tower(l_queue, t_queue, l_dur, t_dur, occupied):
    
    Let the tower direct take-off and landing
    
    Simulates the airport tower allowing planes to take off and land
    according to their queues and runway priority.
    
    Parameters
    ----------
    l_queue : list
        List of planes waiting for landing. The value of each element in the
        list is the time (min) the plane has been in the list.
    t_queue : list
        List of planes waiting for take off. The value of each element in
        the list is the time (min) the plane has been in the list.
    l_dur : int
        Time it takes for plane to land (min)
    t_dur : int
        Time it takes for plane to take-off (min)
    occupied : int
        Timer that keeps track of amount of time runway is occupied for a
        take-off or landing event (min)

    Returns
    -------
    occupied : int
        Updated value of the timer that keeps track of the amount of time
        the runway is occupied for a take-off or landing event (min)
    l_queue_time : int
        Amount of time (min) spent in the queue by the aircraft currently
        landing (0 if no plane lands)
    t_queue_time : int
        Amount of time (min) spent in the queue by the aircraft currently
        taking off (0 if no plane takes off)
    
    
    l_queue_time = 0
    t_queue_time = 0

    # Use the runway if it is available and needed
    if not occupied:
        
        if len(l_queue) > 0:
            # Let planes land first
            l_queue_time, occupied = runway(l_queue, l_dur)
            print('Landing...')

        elif len(t_queue) > 0:
            # Or let planes takeoff
            t_queue_time, occupied = runway(t_queue, t_dur)
            print('Taking off...')
else:
    # Decrease runway activity timer by one minute
    occupied -= 1

    # Return the updated timer and wait times for planes using runway
    return (occupied, l_queue_time, t_queue_time)

def report(t, t_queue, l_queue, occupied):
    
    """
    Report the status of each queue and the runway
    
    Reports the number of planes and their queue wait times for planes in the
    take-off and landing queues together with the simulation time and the
    timer value for the runway activity timer.
    
    Parameters
    ----------
    l_queue : list
        List of planes waiting for landing. The value of each element in the
        list is the time (min) the plane has been in the list.
    t_queue : list
        List of planes waiting for take off. The value of each element in
        the list is the time (min) the plane has been in the list.
    l_dur : int
        Time it takes for plane to land (min)
    t_dur : int
        Time it takes for plane to take-off (min)
    occupied : int
        Timer that keeps track of amount of time runway is occupied for a
        take-off or landing event (min)
    
    Returns
    -------
    lines : str
        String containing the information printed to the screen. This string
        could be used, for instance, to write a file to store the simulation
        results.
    """
    col_widths = [6, 14, 13, 14, 13, 15]
    if t == 0:
        # Print the header information
        cols = ['Time', 'Takeoff Queue', 'Takeoff Wait', 'Landing Queue',
                'Landing Wait', 'Runway Activity']
        units = ['(min)', '(#)', '(min)', '(#)', '(min)', '(min)']
        for i in range(len(cols)):
            cols[i] = cols[i].ljust(col_widths[i])
            units[i] = units[i].center(col_widths[i])
        header = ['|'.join(cols) + '\n']
        header.append('|'.join(units) + '\n')
        header = ''.join(header)
        print(header, end='')
        lines = header
else:
    # Create a list of the current runway state
    data = [t]
    data.append(len(t_queue))
    if len(t_queue) > 0:
        data.append(t_queue[0])
    else:
        data.append(0)
    data.append(len(l_queue))
    if len(l_queue) > 0:
        data.append(l_queue[0])
    else:
        data.append(0)
    data.append(occupied)

    # Create a string of data to print
    for i in range(len(data)):
        data[i] = str(data[i]).center(col_widths[i])
    data = '| '.join(data) + '

    print(data, end='')
lines = data

return lines

def update_stats(num, sum_wait, max_wait, event_wait):
    """
    Update variables used to keep queue and runway statistics

    Update variables needed to keep track of the total number of planes that
    were in a queue, the average wait time, and the maximum wait time.

    Parameters
    ----------
    num : int
        Number of planes that have been in a given queue
    sum_wait : int
        Total duration of time (min) spent by planes in a given queue
    max_wait : int
        Longest time (min) spent by a plane in a given queue
    event_wait : int
        Amount of time (min) spent in the queue by the plane exiting the
        queue. If zero, no member is exiting queue

    Returns
    ------
    num : int
        Number of planes that have been in a given queue
    sum_wait : int
        Total duration of time (min) spent by planes in a given queue
    max_wait : int
        Longest time (min) spent by a plane in a given queue
    """
    # Update the appropriate statistics variables
if event_wait:
    # Member is exiting queue
    num += 1
    sum_wait += event_wait
    if event_wait > max_wait:
        max_wait = event_wait

return (num, sum_wait, max_wait)

def calc_stats(t, t_num, t_wait_sum, t_wait_max, l_num, l_wait_sum, l_wait_max):
    """
    Calculate and print the simulation statistics
    """
    Parameters
    ----------
    t : int
        Total duration of the simulation (min)
    t_num : int
        Total number of take offs
    t_wait_sum : int
        Total duration of time (min) spent by planes in take-off queue
    t_wait_max : int
        Maximum time (min) a plane waited in the take-off queue
    l_num : int
        Total number of landings
    l_wait_sum : int
        Total duration of time (min) spent by planes in landing queue
    l_wait_max : int
        Maximum time (min) a plane waited in the landing queue

    Returns
    -------
    t_wait_mean : float
        Average wait time (min) in take-off queue
    l_wait_mean : float
        Average wait time (min) in landing queue

    """
    sim_dur = t / 60.
    t_wait_mean = t_wait_sum / t_num
    l_wait_mean = l_wait_sum / l_num

    print('Simulation statistics:

Simulation duration (hr) = ', t/60.)
print('Number of take-offs = ', t_num)
print('Average wait in take-off queue %4.4g (min)' % t_wait_mean)
print('Maximum wait in take-off queue %4.4g (min)' % t_wait_max)
print('Number of landings = ', l_num)
print('Average wait in landing queue %4.4g (min)' % l_wait_mean)
print('Maximum wait in landing queue %4.4g (min)' % l_wait_max)

return (t_wait_mean, l_wait_mean)
if __name__ == '__main__':

    # Prepare some test data for one moment during a simulation
    landing_rate = 8
    landing_dur = 3
    takeoff_rate = 8
    takeoff_dur = 2
    t = 36
    landing = []
    takeoff = [4, 3, 2]
    runway_occupied = 0
    landing_time = 0
    takeoff_time = 0
    num_landings = 1
    sum_landing_wait = 3
    max_landing_wait = 3
    num_takeoffs = 6
    sum_takeoff_wait = 19
    max_takeoff_wait = 6

    # Test the report() function
    print('nTesting script for runway_queue.py')
    print('nInitial queue parameters are:
    report(0, takeoff, landing, runway_occupied)
    report(t, takeoff, landing, runway_occupied)

    # Test the arrival() function
    print('nSimulate landing queue arrival over 1 hr period')
    print('Original landing queue is ', landing)
    landing_rate = 8
    n_landing = arrival(landing, landing_rate, delta_t=60)
    print('Number of planes added to landing queue = ', n_landing)
    print('New landing queue is ', landing)

    # Test the waiting() function
    print('nAdd 5 min to the wait queue for take offs')
    print('Original wait queue: ', takeoff)
    waiting(takeoff, delta_t=5)
    print('Updated wait queue: ', takeoff)

    # Test the runway() function
    print('nLet one plane take off')
    print('Planes in queue: ', takeoff)
    takeoff_time, runway_occupied = runway(takeoff, takeoff_dur)
    print('Time required for takeoff (min): ', takeoff_dur)
    print('Time this plane waited to take off (min): ', takeoff_time)
    print('Duration runway will be occupied by this plane (min): ',
          runway_occupied)
    print('Updated takeoff queue: ', takeoff)

    # Test the tower() function
    print('nLet the tower() function select a plane to use the runway')
Reset the runway timer to zero.

runway_occupied = 0

print('Status before running tower():')
report(t, takeoff, landing, runway_occupied)
runway_occupied, landing_time, takeoff_time = tower(landing,
    takeoff, landing_dur, takeoff_dur, runway_occupied)
print('Status after running tower()')
report(t, takeoff, landing, runway_occupied)

if landing_time:
    print('Tower allowed a plane to land.')
elif takeoff_time:
    print('Tower allowed a plane to takeoff.')

print('Time runway will be occupied (min): ', runway_occupied)

# Test the update_stats() function
print('
Update the statistical counters')
print('Tower just allowed a plane to land after waiting')
print('%.4g minutes in the landing queue' % landing_time)
print('Before update:')
print('number of landings = ', num_landings)
print('total landing wait time (min) = ', sum_landing_wait)
print('maximum landing wait time (min) = ', max_landing_wait)
num_landings, sum_landing_wait, max_landing_wait = update_stats(
    num_landings, sum_landing_wait, max_landing_wait, landing_time)
print('After update:')
print('number of landings = ', num_landings)
print('total landing wait time (min) = ', sum_landing_wait)
print('maximum landing wait time (min) = ', max_landing_wait)
print('')
When we execute `runway_queue.py` directly, the test cases report:

Testing script for runway_queue.py
----------------------------------

Initial queue parameters are:

<table>
<thead>
<tr>
<th>Time (min)</th>
<th>Takeoff Queue (#)</th>
<th>Takeoff Wait (min)</th>
<th>Landing Queue (#)</th>
<th>Landing Wait (min)</th>
<th>Runway Activity (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Simulate landing queue arrival over 1 hr period

Original landing queue is []

Number of planes added to landing queue = 8

New landing queue is [47, 29, 28, 20, 8, 6, 3, 2]

Add 5 min to the wait queue for take offs

Original wait queue: [4, 3, 2]

Updated wait queue: [9, 8, 7]

Let one plane take off

Planes in queue: [9, 8, 7]

Time required for takeoff (min): 2

Time this plane waited to take off (min): 11

Duration runway will be occupied by this plane (min): 2

Updated takeoff queue: [8, 7]

Let the `tower()` function select a plane to use the runway

Reset the runway timer to zero.

Status before running `tower()`:

<table>
<thead>
<tr>
<th>Time (min)</th>
<th>Takeoff Queue (#)</th>
<th>Takeoff Wait (min)</th>
<th>Landing Queue (#)</th>
<th>Landing Wait (min)</th>
<th>Runway Activity (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>47</td>
<td>0</td>
</tr>
</tbody>
</table>

Landing...

Status after running `tower()`

<table>
<thead>
<tr>
<th>Time (min)</th>
<th>Takeoff Queue (#)</th>
<th>Takeoff Wait (min)</th>
<th>Landing Queue (#)</th>
<th>Landing Wait (min)</th>
<th>Runway Activity (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>29</td>
<td>3</td>
</tr>
</tbody>
</table>

Tower allowed a plane to land.

Time runway will be occupied (min): 3

Update the statistical counters

Tower just allowed a plane to land after waiting

50 minutes in the landing queue

Before update:
number of landings = 1
total landing wait time (min) = 3
maximum landing wait time (min) = 3

After update:
number of landings = 2
total landing wait time (min) = 53
maximum landing wait time (min) = 50
Listing for runway_sim.py

# runway_sim.py
#
# Simulate take-off and landing queues for a single runway.
#
# Simulate queues of airplanes waiting to take-off and landing at a single
# runway. Uses functions for arrival times and runway events in the module
# runway_queue.
#
# S. Socolofsky
# ENGR 102
# October 2018

import runway_queue

# Introduce program
print('
Runway Simulator

Simulates take-off and landing queues for a single runway
The simulation uses a time-step of one minute
and updates the take-off and landing queues each minute while also simulating runway activity.

# Get the parameters of the airport and runway
takeoff_dur, landing_dur, takeoff_rate, landing_rate, sim_dur = 
    runway_queue.get_sim_parameters()

# Initialize the take-off and landing queues
landing = []
takeoff = []

# Simulate the runway in one-minute intervals
for i in range(int(sim_dur * 60)):
    # Get new planes arriving for landing and take-off
    runway_queue.arrival(landing, landing_rate)
    runway_queue.arrival(takeoff, takeoff_rate)
# Ask the tower to use the runway to land or take-off

\[
\text{(runway\_occupied, landing\_time, takeoff\_time)} = \text{runway\_queue.tower(}
\]
landing, takeoff, landing\_dur, takeoff\_dur, runway\_occupied
\]

# Update wait times and statistics

\[
t += 1
\]
runway\_queue.waiting\(\text{takeoff}\)
runway\_queue.waiting\(\text{landing}\)

\[
\text{num\_landings, sum\_landing\_wait, max\_landing\_wait} = \backslash
\]
runway\_queue.update\_stats\(\text{num\_landings, sum\_landing\_wait, max\_landing\_wait, landing\_time}\)

\[
\text{num\_takeoffs, sum\_takeoff\_wait, max\_takeoff\_wait} = \backslash
\]
runway\_queue.update\_stats\(\text{num\_takeoffs, sum\_takeoff\_wait, max\_takeoff\_wait, takeoff\_time}\)

# Print runway status

runway\_queue.report\(t, takeoff, landing, runway\_occupied\)

# Report the statistics of the runway simulation.

takeoff\_wait\_mean, landing\_wait\_mean = \text{runway\_queue.calc\_stats(t, takeoff\_time, landing\_time, num\_landings, num\_takeoffs, sum\_takeoff\_wait, max\_takeoff\_wait, sum\_landing\_wait, max\_landing\_wait)}
Runway Simulator

Simulates take-off and landing queues for a single runway

The simulation uses a time-step of one minute
and updates the take-off and landing queues each
minute while also simulating runway activity.

Enter the parameter values of the runway simulation:
- Time required to take off (min): 2
- Time required to land (min): 3
- Average number of planes taking off per hour: 8
- Average number of planes landing per hour: 8

Enter the number of hours to simulate: 1

<table>
<thead>
<tr>
<th>Time (min)</th>
<th>Takeoff Queue (#)</th>
<th>Takeoff Wait (min)</th>
<th>Landing Queue (#)</th>
<th>Landing Wait (min)</th>
<th>Runway Activity (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Taking off...
- 9 | 0 | 0 | 0 | 0 | 2.0
- 10 | 0 | 0 | 1 | 1 | 1.0
- 11 | 0 | 0 | 1 | 2 | 0.0

Landing...
- 12 | 0 | 0 | 0 | 0 | 3.0
- 13 | 0 | 0 | 0 | 0 | 2.0
- 14 | 0 | 0 | 1 | 1 | 1.0
- 15 | 0 | 0 | 1 | 2 | 0.0

Landing...
- 16 | 0 | 0 | 0 | 0 | 3.0
- 17 | 0 | 0 | 1 | 1 | 2.0
- 18 | 0 | 0 | 1 | 2 | 1.0
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Taking off...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Taking off...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Taking off...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Landing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>2</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>2</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Taking off...
53 | 1 | 8 | 0 | 0 | 2.0
54 | 1 | 9 | 1 | 1 | 1.0
55 | 1 | 10 | 2 | 2 | 0.0
Landing...
56 | 2 | 11 | 1 | 2 | 3.0
57 | 2 | 12 | 1 | 3 | 2.0
58 | 2 | 13 | 1 | 4 | 1.0
59 | 2 | 14 | 1 | 5 | 0.0
Landing...
60 | 2 | 15 | 0 | 0 | 3.0

Simulation statistics:
Simulation duration (hr) = 1.0

Number of take-offs = 5
Average wait in take-off queue 9.8 (min)
Maximum wait in take-off queue 16 (min)

Number of landings = 10
Average wait in take-off queue 5.9 (min)
Maximum wait in take-off queue 9 (min)