Kushal Kale ksk7657@rit.edu
Vedika Vishwanath Painjane vp2312@rit.edu
Arjun Kozhissery ak8913@rit.edu

```
class Passenger:
        __slots__ = “name”, “ticket_number”, “has_carry_on”
        name: str
        ticket_number: str
        has_carry_on: boolean

        def __init__(self, name: str, ticket_number: str, has_carry_on: boolean = False):
                self.name = name
                self.ticket_number = ticket_number
                self.has_carry_on  = has_carry_on

        def __str__(self):
                has_carry_on = “has carry on luggage” if self.has_carry_on else “doesn’t have carry on luggage”
                return self.name + “ (” + self.ticket_number + “) ” + has_carry_on
```

2. Class Gate

```
Class Gate:
    __slots__ = “current_pax_count”, “boarding_zones”, “max_limit”


    def __init__(self, max_limit, current_pax_count = 0 ):
        for i in range(4):
            self.boarding_zones.append( Queue(maxsize = self.max_limit))
        self.max_limit = max_limit
        self.current_pax_count = current_pax_count

   Def add_passenger( passenger, boarding_zone ):
   “””
   Adds a passenger to the specified boarding zone in the gate
   ”””
   If self.current_pax_count + 1 >  self.max_limit or not (boarding_Zone >= 1 and boarding_zone <=4):
       # adding this passenger would violate the requirement or invalid boarding zone passed
                Return false
```

```
self.Boarding_zones[ boarding_zone - 1].put(passenger)
self.current_pax_count += 1
If current_pax_count == self.max_limit :
        Return false
Return true;
```

a. 4 Queues, we need a counter to keep track of the number of passengers queued up. This can be done by keeping the counter as a member of the Gate class. This counter would be incremented every time an enqueue operation is called on either of the four queues and decremented whenever pop is called.
b. Before adding passenger to the gate, check counter with the maximum number of passengers allowed, if the counter is greater than the maximum number of passengers, then stop intake of passengers.

3.
```
func queue_pax(passenger, gate) {
    Initialize zone
    current_pax_count = gate.current_pax_count
    Store queues in boarding_zone_array

    If this.current_pax_count + 1 > self.max_limit:
        Return false
    else:
        boarding_zone_array[zone-1].put(passenger)
        this.current_pax_count = this.current_pax_count + 1
        If this.current_pax_count == gate.max_limit:
            Return False
        Else:
            Return True
```

Use the value of zone to get the queue corresponding to the boarding zone (internally, the queues will be stored in an array and the index of the queue corresponding to zone ‘n’ would be n-1)

Try to queue the passenger using the method defined in the gate class. This method would return false when adding th

```
}
```

4. The plane would have 2 stacks, one for passengers with carry-on and one for passengers without carry-on.(Python lists can be used)

```
5. function(gate, aircraft)
        for i in range(4):
        bz = gate.boarding_zones[3-i]
        while not bz.empty():
                pax = bz.get()
                if (aircraft is not full):
                        aircraft.push(pax)
```