```
func simulate_mission(list of original_missions):
     assume there are missions in the missions list
     missions = copy of original_missions list
     result = []

     sort the missions in reverse order based on their day of
arrival
     day = day of arrival of the last mission in the list
     heap = popped missions of the same group based on day of
arrival
     heapify(heap)

     while there are missions in the heap:
          if there are missions left and day == day of arrival of
          the last mission:
               For each mission in the popped missions of the same
               group based on day of arrival:
                    Push the popped missions into the heap while
                    maintaining the min heap structure

          current_mission = pop the mission with the shortest length
          from the heap

          if first_execution of current_mission == -1:
               first_execution of current_mission = day

          remaining days of current_mission -= 1

          if remaining days of current_mission > 0:
               push the current_mission to the heap
          else:
               completion day of current_mission = day + 1
               for each original_mission in original_missions:
                    if name of original_mission == name of
                    current_mission:
                         completion day of original_mission =
                         completion day of current_mission
                         first execution day of original_mission =
                         first execution day of current_mission
                         break
          add current_mission name to result[]
          day += 1
     return result, original_missions
```