In [ ]:
```
NAME = "Esther"
COLLABORATORS = "Ha, Kalyane"
```

In [ ]:
```
from google.colab import files
from IPython.display import Image
```

In [ ]:
```
upload = files.upload()
```

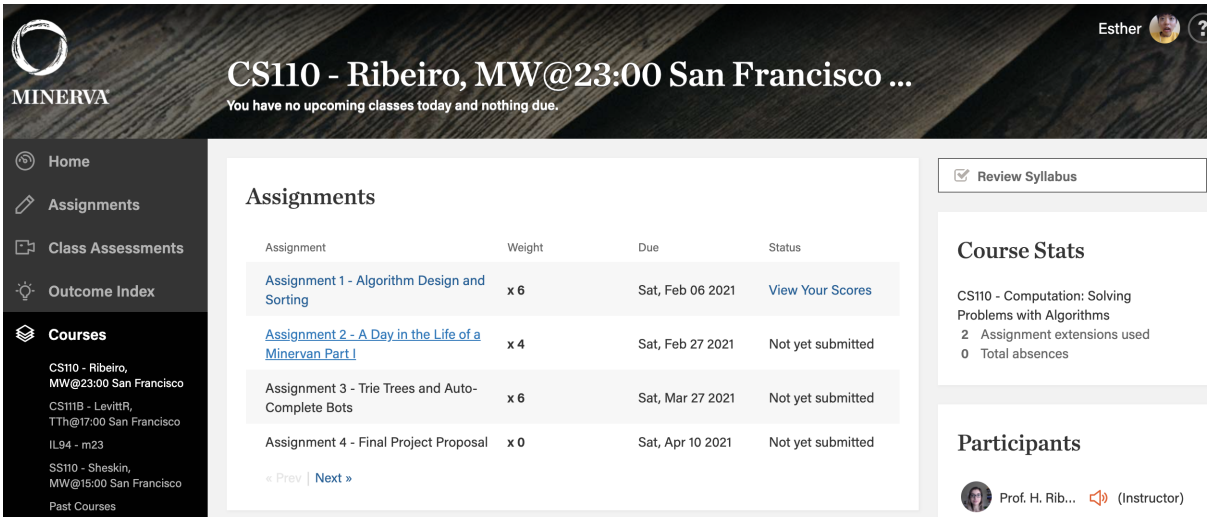
Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Screen Shot 2021-02-28 at 10.29.54 AM.png to Screen Shot 2021-02
-28 at 10.29.54 AM.png
```

In [ ]:
```
Image('Screen Shot 2021-02-28 at 10.29.54 AM.png')
```

Out[ ]:



Q1 [#AlgorithmicStrategies, #DataStructures]

Prepare a table containing all the activities that you plan to do in the city of your rotation

id (task-subtask)	description	duration (min)	dependencies	happiness score
1	Go to CS 111B class 10:00AM	90	no	2
2	Go back to bed and take a nap	60	1	8
3-1	Go hiking in Namsan Mountain -- Wait Sam in the lobby	10	2	7
3-2	Go hiking in Namsan Mountain -- Run all the staircases in the mountain	70	3-1	7
3-3	Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain	40	3-2	7
3-4	Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset	30	3-3	7
4-1	Play table tennis in the res -- Find people who want to play too	30	2	5
4-2	Play table tennis in the res -- Play a round with Gabrial	15	4-1	5
4-3	Play table tennis in the res -- Play two rounds with Nikita who plays tennis	15	4-1	6
5-1	Get dinner with Mariia in haebang chon -- Change to warm clothes	5	3,4	6
5-2	Get dinner with Mariia in haebang chon -- Climb upstairs	10	5-1	4
5-3	Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants	10	5-2	4
5-4	Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken	15	5-3	4
5-5	Get dinner with Mariia in haebang chon -- Take it back to the residential hall and eat together	10	5-4	3
6	Discuss CS 11B assignment in my flatmates	60	1	1
7-1	Late night sight seeing with under Seoul tower -- Climb up in the woods while chatting with Yufei	50	5	9
7-2	Late night sight seeing with under Seoul tower -- Reach the top, and appreciate the night view of Seoul	50	7-1	9
7-3	Late night sight seeing with under Seoul tower -- Go down	50	7-2	9
7-4	Late night sight seeing with under Seoul tower -- Buy Soju and chocolate	50	7-3	9
8-1	Ready for bed - Shower	20	7-4	5
8-2	Go home and get ready for bed - Brush teeth	5	8-1	5
8-3	Go home and get ready for bed - Say goodnight to roommate Dodie and rest in peace	10	8-2	8

How will you store information about these activities and sub-tasks? I will store the information in a nested dictionary, where the id is the key of the outer dictionary, and its value includes description, duration, and dependencies.

Describe how your scheduler will work, with an emphasis on why a priority queue is a well-suited data structure to handle the prioritization of tasks, and how you have defined and computed the priority value of each task and/or sub-task.

A. Different functions: heap class, check if there are unscheduled tasks, find tasks without dependencies, push the task into priority queue, remove_dependency, task remaining time update.

B. Information storage: I use a nested dictionary to store my tasks. Since we have lots of variables in the scheduler, using nested list is likely to confuse readers. For instance, when calling a list, I can specificy task[duration] for the time length of a task in a dictionary, rather than task[3] in a list.

C: My scheduler is based on 2 criteria.

1. Dependency
2. Happiness Score

First, my scheduler will extract all independent task through get_ready_tsks, and then prioritize those tasks based on happiness score. Second, I use happiness score to prioritize tasks. Using Min Heap is because I like to do the hardest things first(lowest happiness score), so I can enjoy what I love to do later on.

Namsan Mountain Hiking, Soju, and Seoul night view

Q2 [#PythonProgramming, #CodeReadability]

Program an activity scheduler in Python, which receives the list of tasks above as input and returns a schedule for you to follow. Please refrain from using any external Python library besides pandas, math, and random modules (if you plan on using other libraries, please check with your course instructor first).

Make sure your internal representation of tasks has all the fields described in Figure 1, in addition to the priority value that will characterize each task. Your activity scheduler must report at the end of every timestep that a task has been completed. The program ends when all tasks have been completed.

In [ ]:

```
tasks = { 1:{'description':'Go to CS 111B class 10:00AM', 'duration':90,
'dependency':[], 'status': 'N', 'score':2},
          2:{'description':'Go back to bed and take a nap', 'duration':6
0, 'dependency':[1], 'status': 'N', 'score':8},
          3.1:{'description':'Go hiking in Namsan Mountain -- Wait Sam i
n the lobby', 'duration':10, 'dependency':[2], 'status': 'N', 'score': 7
},
         3.2:{'description':'Go hiking in Namsan Mountain -- Run all the
staircases in the mountain', 'duration':70, 'dependency':[3.1], 'status'
: 'N','score':7},
         3.3:{'description':'Go hiking in Namsan Mountain -- Workout in
 the outdoor gym on the top of the mountain', 'duration':40, 'dependenc
y':[3.2], 'status': 'N','score':7},
         3.4:{'description':'Go hiking in Namsan Mountain -- Walk down t
he mountain and enjoy the sunset', 'duration':30, 'dependency':[3.3], 's
tatus': 'N','score':7},
         4.1 :{'description':'Play table tennis in the res -- Find peopl
e who want to play too', 'duration':30, 'dependency':[2], 'status': 'N',
'score':5},
         4.2 :{'description':'Play table tennis in the res -- Play a rou
nd with Gabrial', 'duration':15, 'dependency':[4.1], 'status': 'N','scor
e':5},
         4.3 :{'description':'Play table tennis in the res -- Play two r
ounds with Nikita who plays tennis', 'duration':15, 'dependency':[4.1],
'status': 'N','score':6},
         5.1 :{'description':'Get dinner with Mariia in haebang chon --
 Change to warm clothes', 'duration':5, 'dependency':[3.1, 3.2,3.3, 3.4,
4.1, 4.2,4.3], 'status': 'N','score':6},
         5.2 :{'description':'Get dinner with Mariia in haebang chon --
 Climb upstairs', 'duration':10, 'dependency':[5.1], 'status': 'N','scor
e':4},
         5.3 :{'description':'Get dinner with Mariia in haebang chon --
 Walk to the street full of grocery stores, street food, and restaurant
s', 'duration':10, 'dependency':[5.2], 'status': 'N','score':4},
         5.4 :{'description':'Get dinner with Mariia in haebang chon --
 Wait for Korean orange chicken', 'duration':15, 'dependency':[5.3], 'st
atus': 'N','score':4},
         5.5 :{'description':'Get dinner with Mariia in haebang chon --
 Take it back to the residential hall and eat together', 'duration':10,
'dependency':[5.4], 'status': 'N','score':3},
         6 :{'description':'Discuss CS 11B assignment in my flatmates',
'duration':60, 'dependency':[2], 'status': 'N','score':1},
         7.1 :{'description':'Late night sight seeing with under Seoul t
ower -- Climb up in the woods while chatting with Yufei', 'duration':50,
'dependency':[5.1, 5.2,5.3,5.4], 'status': 'N','score':9},
         7.2 :{'description':'Late night sight seeing with under Seoul t
ower -- Reach the top, and appreciate the night view of Seoul', 'duratio
n':50, 'dependency':[7.1], 'status': 'N','score':9},
         7.3 :{'description':'Late night sight seeing with under Seoul t
ower -- Go down', 'duration':50, 'dependency':[7.2], 'status': 'N','scor
e':9},
         7.4 :{'description':'Late night sight seeing with under Seoul t
ower -- Buy Soju and chocolate', 'duration':50, 'dependency':[7.3], 'sta
tus': 'N','score':9},
         8.1 :{'description':'Ready for bed  - Shower', 'duration':20,
'dependency':[7.4], 'status': 'N','score':5},
```

```
        8.2 :{'description':'Ready for bed - Brush teeth', 'duration':5
, 'dependency':[8.1], 'status': 'N','score':5},
        8.3 :{'description':'Ready for bed - Say goodnight to roommate
 Dodie and rest in peace', 'duration':10, 'dependency':[8.2], 'status':
'N','score':8},
        }
```

In [ ]:

```
def print_input_tasks(tasks):
    """
    Input: list of tasks
    Task Status:
    - 'N' : Not yet in priority queue (default status)
    - 'I' : In priority queue
    - 'C' : Completed
    Output: print statement with all the tasks to be included in the sch
eduler
    """
    print('Input List of Tasks')
    for id, value in tasks.items():
        print(f"task:{id} \t {value['description']} \t duration:{value
['duration']} \t depends on: {value['dependency']} \t Status: {value['st
atus']} \t Happiness: {value['score']}")

print_input_tasks(tasks)
```

```
Input List of Tasks
task:1   Go to CS 111B class 10:00AM     duration:90     depends on: []
Status: N        Happiness: 2
task:2   Go back to bed and take a nap   duration:60     depends on:
[1]      Status: N       Happiness: 8
task:3.1         Go hiking in Namsan Mountain -- Wait Sam in the lobby
duration:10      depends on: [2]         Status: N       Happiness: 7
task:3.2         Go hiking in Namsan Mountain -- Run all the staircases
in the mountain          duration:70     depends on: [3.1]       Statu
s: N     Happiness: 7
task:3.3         Go hiking in Namsan Mountain -- Workout in the outdoor
gym on the top of the mountain   duration:40     depends on: [3.2]
Status: N        Happiness: 7
task:3.4         Go hiking in Namsan Mountain -- Walk down the mountain
and enjoy the sunset     duration:30     depends on: [3.3]       Statu
s: N     Happiness: 7
task:4.1         Play table tennis in the res -- Find people who want t
o play too       duration:30     depends on: [2]         Status: N
Happiness: 5
task:4.2         Play table tennis in the res -- Play a round with Gabr
ial      duration:15     depends on: [4.1]       Status: N       Happin
ess: 5
task:4.3         Play table tennis in the res -- Play two rounds with N
ikita who plays tennis   duration:15     depends on: [4.1]       Statu
s: N     Happiness: 6
task:5.1         Get dinner with Mariia in haebang chon -- Change to wa
rm clothes       duration:5      depends on: [3.1, 3.2, 3.3, 3.4, 4.1,
4.2, 4.3]        Status: N       Happiness: 6
task:5.2         Get dinner with Mariia in haebang chon -- Climb upstai
rs       duration:10     depends on: [5.1]       Status: N       Happin
ess: 4
task:5.3         Get dinner with Mariia in haebang chon -- Walk to the
street full of grocery stores, street food, and restaurants      durati
on:10    depends on: [5.2]       Status: N       Happiness: 4
task:5.4         Get dinner with Mariia in haebang chon -- Wait for Kor
ean orange chicken       duration:15     depends on: [5.3]       Statu
s: N     Happiness: 4
task:5.5         Get dinner with Mariia in haebang chon -- Take it back
to the residential hall and eat together         duration:10     depend
s on: [5.4]      Status: N       Happiness: 3
task:6   Discuss CS 11B assignment in my flatmates       duration:60
depends on: [2]          Status: N       Happiness: 1
task:7.1         Late night sight seeing with under Seoul tower -- Clim
b up in the woods while chatting with Yufei      duration:50     depend
s on: [5.1, 5.2, 5.3, 5.4]       Status: N       Happiness: 9
task:7.2         Late night sight seeing with under Seoul tower -- Reac
h the top, and appreciate the night view of Seoul        duration:50
depends on: [7.1]        Status: N       Happiness: 9
task:7.3         Late night sight seeing with under Seoul tower -- Go d
own      duration:50     depends on: [7.2]       Status: N       Happin
ess: 9
task:7.4         Late night sight seeing with under Seoul tower -- Buy
Soju and chocolate       duration:50     depends on: [7.3]       Statu
s: N     Happiness: 9
task:8.1         Ready for bed  - Shower         duration:20     depend
s on: [7.4]      Status: N       Happiness: 5
task:8.2         Ready for bed - Brush teeth     duration:5      depend
```

```
s on: [8.1]      Status: N       Happiness: 5
task:8.3         Ready for bed - Say goodnight to roommate Dodie and re
st in peace      duration:10     depends on: [8.2]       Status: N
Happiness: 8
```

In [ ]:
```python
#Check if any unscheduled task exists
def unscheduled_tasks(tasks):
    """
    Input: list of tasks
    Output: boolean (checks the status of all tasks and returns `True` i
f at least one task has status = 'N')
    """
    for id, task in tasks.items():
        if task['status'] == 'N':
            return True
    return False
```

In [ ]:
```python
#Remove the dependencies if we complete the task

def remove_dependency(tasks, complete_task):
    """
    Input: list of tasks and task_id of the task just completed
    Output: lists of tasks with t_id removed
    """

    for id, task in tasks.items():
      if id != complete_task and complete_task in task['dependency']:
        task['dependency'].remove(complete_task) #remove the complete ta
sk from the dependency list
```

In [ ]:
```python
#Find the task we can execute right now(without dependency and we have
n't done it yet)
def get_ready_tsks(tasks):
    """
    Implements step 1 of the scheduler
    Input: list of tasks
    Output: list of tasks that are ready to execute (i.e. tasks with no
 pendending task dependencies)
    """
    ready_task = []

    for id, task in tasks.items():
      if task['status'] == 'N' and not task['dependency']: # If tasks ha
s no dependencies and is not yet in queue
        task['status'] = 'I' # Change status of the task
        ready_task.append((task['score'], {'id': id}, {'duration': task[
'duration']})) # If tasks has no dependencies and is not yet in queue

    return ready_task
# get_ready_tsks(tasks)
```

In [ ]:

```
def left(i):              # left(i): takes as input the array index of a
 parent node in the binary tree and
    return 2*i + 1        #          returns the array index of its left
 child.

def right(i):             # right(i): takes as input the array index of a
parent node in the binary tree and
    return 2*i + 2        #           returns the array index of its righ
t child.

def parent(i):            # parent(i): takes as input the array index of
 a node in the binary tree and
    return (i-1)//2       #            returns the array index of its par
ent



class MinHeapq:

  def __init__(self):
    #initiate the size and empty list to store the sorted element
    self.size = 0
    self.heap = []

  def heappush(self,key):  #push the nodes in the empty list
    self.heap.append(-float("inf")) #create an empty space with negative
infinity
    self.size += 1
    self.decrease_key(self.size, key)

  def decrease_key(self, i, key):
    i = i-1       #index = len(heap) - 1
    self.heap[i] = key
    while i > 0 and self.heap[parent(i)] > self.heap[i]:
      self.heap[parent(i)], self.heap[i] = self.heap[i], self.heap[paren
t(i)]
      i = parent(i)

  def heapify(self, i):

    l = left(i)
    r = right(i)
    heap = self.heap

    # if there is a left child node for current node and the child is bi
gger, we set the a new smallest index
    # l < len(heap) to prevent the index exceed the length of the list
    if (l < len(heap)) and (heap[l] < heap[i]):
      smallest = l
    else:
      smallest = i

    # if there is a left child node for current node and the child is bi
gger, and compare with the new smallest
    if (r < len(heap)) and (heap[r] < heap[smallest]):
        smallest = r
```

```
    #the index of the smallest node is not the parent node, swap the sma
llest node with its parent
    if smallest != i:
      heap[i], heap[smallest] = heap[smallest], heap[i]
      self.heapify(smallest)
    #recursively return the heapify process from the smallest index
    return heap

  def mink(self): #Find the min value(the root node)
     return self.heap[0]

  def heappop(self):
    if self.size < 1:
      raise ValueError('Heap underflow: There are no keys in the priorit
y queue ')

    min = self.heap[0] #find the min ??????????????????????can I use min
k?????
    self.heap[0] = self.heap[-1] #replace the root node with the leaf no
de with biggest index, but it didn't delete the last value
    self.heap.pop() #delete the leaf node
    self.size-=1
    self.heapify(0)
    return min
```

Using Min Heap is because I like to do the hardest things first(lowest happiness score), so I can enjoy what I love to do later on.

In [ ]:
```
def task_time_update(task_time):
  """
    Implements time step calcluation for the scheduler
    Input: total task time
    Output: remaining task time after a time step
    """
  global c_time
  step = step_size #step is the scheduler's time step
  if task_time <= step: #If it is less than the step_size take a smaller
time step
    step = task_time
  task_time -= step # STEP 5: adjust the tasks remaining time
  c_time += step
  return task_time
```

In [ ]:
```
# Inputs Parameters to the Scheduler
step_size = 5  # 5 mintues as a step size
c_time = 600  # current time is set to the initial time in minutes (10:0
0 AM = 10x60)
start = [] #a list to store tasks we have started

pqueue = MinHeapq()

#heapify the ready tasks into priority queue
def add_tasks_pqueue(ready_task):

  for key in ready_task:
      pqueue.heappush(key) #push the tasks into a heap format

  return pqueue.heap

# add_tasks_pqueue(ready_task)
```

In [ ]:
```
def Start_printer(id_dic): #print out when starting the task
  global start
  if id_dic['id'] not in start:
    start.append(id_dic['id'])
    return True
  else:
    return False
```

In [ ]:

```
while unscheduled_tasks(tasks) or pqueue.heap: #if there are unscheduled
tasks or ready tasks in the wait list
  #STEP 1: Extract  tasks ready to execute (those without dependencies)
  ready_task = get_ready_tsks(tasks)


  #STEP 2: Push the tasks onto the priority queue
  add_tasks_pqueue(ready_task)

  if pqueue.heap:
    score, id_dic, duration_dic = pqueue.heappop()

    #Check if we have started this task before
    if Start_printer(id_dic):
      print('⭐ Start task:',id_dic['id'], tasks[id_dic['id']]['descript
ion'], 'at', int(c_time/60) ,':',int((c_time) % 60))

    #STEP 3: Execute the main task
    remain_time = task_time_update(duration_dic['duration'])
    if remain_time == 0: #if we finish the task, we remove the dependenc
y
      print('✅ Complete task:',id_dic['id'], tasks[id_dic['id']]['descr
iption'],'at 🕒 ', int(c_time/60) ,':',int((c_time) % 60))
      remove_dependency(tasks, id_dic['id'])
    else: #if not, push the remaining task  back to task list
      # print(score, id_dic, remain_time)
      pqueue.heappush((score, id_dic, {'duration':remain_time}))
```

⭐ Start task: 1 Go to CS 111B class 10:00AM at 10 : 0
✅ Complete task: 1 Go to CS 111B class 10:00AM at 🕒 11 : 30
⭐ Start task: 2 Go back to bed and take a nap at 11 : 30
✅ Complete task: 2 Go back to bed and take a nap at 🕒 12 : 30
⭐ Start task: 6 Discuss CS 11B assignment in my flatmates at 12 : 30
✅ Complete task: 6 Discuss CS 11B assignment in my flatmates at 🕒 13 : 30
⭐ Start task: 4.1 Play table tennis in the res -- Find people who want to play too at 13 : 30
✅ Complete task: 4.1 Play table tennis in the res -- Find people who want to play too at 🕒 14 : 0
⭐ Start task: 4.2 Play table tennis in the res -- Play a round with Gabrial at 14 : 0
✅ Complete task: 4.2 Play table tennis in the res -- Play a round with Gabrial at 🕒 14 : 15
⭐ Start task: 4.3 Play table tennis in the res -- Play two rounds with Nikita who plays tennis at 14 : 15
✅ Complete task: 4.3 Play table tennis in the res -- Play two rounds with Nikita who plays tennis at 🕒 14 : 30
⭐ Start task: 3.1 Go hiking in Namsan Mountain -- Wait Sam in the lobby at 14 : 30
✅ Complete task: 3.1 Go hiking in Namsan Mountain -- Wait Sam in the lobby at 🕒 14 : 40
⭐ Start task: 3.2 Go hiking in Namsan Mountain -- Run all the staircases in the mountain at 14 : 40
✅ Complete task: 3.2 Go hiking in Namsan Mountain -- Run all the staircases in the mountain at 🕒 15 : 50
⭐ Start task: 3.3 Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain at 15 : 50
✅ Complete task: 3.3 Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain at 🕒 16 : 30
⭐ Start task: 3.4 Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset at 16 : 30
✅ Complete task: 3.4 Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset at 🕒 17 : 0
⭐ Start task: 5.1 Get dinner with Mariia in haebang chon -- Change to warm clothes at 17 : 0
✅ Complete task: 5.1 Get dinner with Mariia in haebang chon -- Change to warm clothes at 🕒 17 : 5
⭐ Start task: 5.2 Get dinner with Mariia in haebang chon -- Climb upstairs at 17 : 5
✅ Complete task: 5.2 Get dinner with Mariia in haebang chon -- Climb upstairs at 🕒 17 : 15
⭐ Start task: 5.3 Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants at 17 : 15
✅ Complete task: 5.3 Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants at 🕒 17 : 25
⭐ Start task: 5.4 Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken at 17 : 25
✅ Complete task: 5.4 Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken at 🕒 17 : 40
⭐ Start task: 5.5 Get dinner with Mariia in haebang chon -- Take it back to the residential hall and eat together at 17 : 40
✅ Complete task: 5.5 Get dinner with Mariia in haebang chon -- Take it back to the residential hall and eat together at 🕒 17 : 50

```
⭐ Start task: 7.1 Late night sight seeing with under Seoul tower -- Cl
imb up in the woods while chatting with Yufei at 17 : 50
✅ Complete task: 7.1 Late night sight seeing with under Seoul tower --
Climb up in the woods while chatting with Yufei at 🕒 18 : 40
⭐ Start task: 7.2 Late night sight seeing with under Seoul tower -- Re
ach the top, and appreciate the night view of Seoul at 18 : 40
✅ Complete task: 7.2 Late night sight seeing with under Seoul tower --
Reach the top, and appreciate the night view of Seoul at 🕒 19 : 30
⭐ Start task: 7.3 Late night sight seeing with under Seoul tower -- Go
down at 19 : 30
✅ Complete task: 7.3 Late night sight seeing with under Seoul tower --
Go down at 🕒 20 : 20
⭐ Start task: 7.4 Late night sight seeing with under Seoul tower -- Bu
y Soju and chocolate at 20 : 20
✅ Complete task: 7.4 Late night sight seeing with under Seoul tower --
Buy Soju and chocolate at 🕒 21 : 10
⭐ Start task: 8.1 Ready for bed  - Shower at 21 : 10
✅ Complete task: 8.1 Ready for bed  - Shower at 🕒 21 : 30
⭐ Start task: 8.2 Ready for bed - Brush teeth at 21 : 30
✅ Complete task: 8.2 Ready for bed - Brush teeth at 🕒 21 : 35
⭐ Start task: 8.3 Ready for bed - Say goodnight to roommate Dodie and
rest in peace at 21 : 35
✅ Complete task: 8.3 Ready for bed - Say goodnight to roommate Dodie a
nd rest in peace at 🕒 21 : 45
```

In addition to the actual scheduler, provide at least one simple example to demonstrate how your scheduler prioritizes tasks based on their priority value.

Example: After completing task 1 and 2. My priority queue has 3 tasks(3.1,4.1,6) that I can execute. Given task 6 has the lowest score, I execute task 6(writing math assignment) first, so I can enjoy task 3.1, 4.1(go hiking, play table tennis) later on.

Q3 [#AlgorithmicStrategies]

Now, you realize that some of the tasks in your schedule can be multi-tasked! In other words, many of your daily tasks can be performed simultaneously (e.g. sipping a local beverage while chatting with a friend at a cafe, taking pictures while riding a bus, or walking in a park). You will modify your algorithmic approach to now handle multi-tasks. Notice that while some tasks can be multi-tasked, others may demand your full attention (e.g. CS110 pre-class work).

Criteria for scheduling non-multi-tasking activities—For tasks that require your full attention, your program cannot schedule any other tasks while that task's status is in_progress. Note that if such a task (say A) is part of the dependencies of another task (say B), then once A's status is set to completed, you will be required to update that property on the dependencies of B and the corresponding priorities.

Criteria for scheduling Multitasking tasks—For tasks that can be done at the same time, they may have a different duration and terminate at different times. As such, you will need to keep track of the remaining time for every task being processed in multi-tasking mode and update the scheduler clock accordingly. In fact, when two or more multi-tasking activities are being executed, the remaining time of every activity needs to be adjusted at the same time. Notice that multi-tasking activities can but need not be executed at the same time. For example, 'eat a delicious pizza' and 'brushing your teeth' don't make sense together, but you can 'eat a pizza' while'talking to a friend', and you can 'brush your teeth' while 'listening to a podcast'. How do you include these constraints in your schedule? Your response to the points below needs to address this.

A. Describe as clearly as you can any changes you will need to make to the first version of the scheduler to include multi-tasking activities.

Three modifications:

1. Decrease the numbers of dependency so I can multi-task.
2. Fix some tasks on a specific schedule e.g. CS 111B task
3. Create a multi tasks list. For instance, I can execute 4.1 (Find people to play table tennis) while executing 1 (taking CS 111B class). I add their ids into each other's multi tasking lists.

B. Describe how constraints in the scheduling process are handled by a priority queue. To implement multi-tasking, I pop out the task, check if it is multi-taskable.

- If yes, we pop out the next candidate task in the priority queue and check it can be done with the main task. If it is, we implement multi-tasking. If it not, we push this candidate task back to the queue. I check only two candidate tasks to save time.
- If not, we execute the main task.

id (task-subtask)	description	duration (min)	dependencies	happiness score	multi_tasking
1	Go to CS 111B class	90	no	2	6, 4-1, 4-2, 4-3
2	Go back to bed and take a nap	60	1	8	
3-1	Go hiking in Namsan Mountain -- Wait Sam in the lobby	10	2	7	
3-2	Go hiking in Namsan Mountain -- Run all the staircases in the mountain	70	3-1	7	
3-3	Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain	40	3-2	7	
3-4	Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset	30	3-3	7	
4-1	Play table tennis in the res -- Find people who want to play too	30	no	5	
4-2	Play table tennis in the res -- Play a round with Gabrial	15	4-1	5	
4-3	Play table tennis in the res -- Play two rounds with Nikita who plays tennis	15	4-1	6	
5-1	Get dinner with Mariia in haebang chon -- Change to warm clothes	5	3,4	6	
5-2	Get dinner with Mariia in haebang chon -- Climb upstairs	10	5-1	4	
5-3	Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants	10	5-2	4	
5-4	Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken	15	5-3	4	
5-5	Get dinner with Mariia in haebang chon -- Take it back to the residential hall and eat together	10	5-4	3	
6	Discuss CS 11B assignment in my flatmates	60	no	1	8-2
7-1	Late night sight seeing with under Seoul tower -- Climb up in the woods while chatting with Yufei	50	5	9	
7-2	Late night sight seeing with under Seoul tower -- Reach the top, and appreciate the night view of Seoul	50	7-1	9	
7-3	Late night sight seeing with under Seoul tower -- Go down	50	7-2	9	
7-4	Late night sight seeing with under Seoul tower -- Buy Soju and chocolate	50	7-3	9	
8-1	Ready for bed - Shower	20	7-4	5	
8-2	Go home and get ready for bed - Brush teeth	5	8-1	5	6
8-3	Go home and get ready for bed - Say goodnight to roommate Dodie and rest in peace	10	8-2	8	

Q4 [#PythonProgramming, #CodeReadability]

Write an activity priority scheduler with multi-tasking capability in Python, which receives as input a list of tasks as outlined in Figure 2, and reports (outputs) a schedule for you to follow. As before, please refrain from using any external Python library besides the math and random module (if you intend on using other libraries, please check with your course instructor first). You can follow the example code for the input and output quoted in Q2.

In [ ]:

```
new_tasks = { 1:{'description':'Go to CS 111B class', 'duration':90, 'de
pendency':[], 'status': 'N', 'score':2, 'multi_task':[6,4.1,4.2,4.3]},
          2:{'description':'Go back to bed and take a nap', 'duration':6
0, 'dependency':[1], 'status': 'N', 'score':8, 'multi_task':[]},
          3.1:{'description':'Go hiking in Namsan Mountain -- Wait Sam i
n the lobby', 'duration':10, 'dependency':[2], 'status': 'N', 'score': 7
,'multi_task':[]},
          3.2:{'description':'Go hiking in Namsan Mountain -- Run all the
staircases in the mountain', 'duration':70, 'dependency':[3.1], 'status'
: 'N','score':7,'multi_task':[]},
          3.3:{'description':'Go hiking in Namsan Mountain -- Workout in
 the outdoor gym on the top of the mountain', 'duration':40, 'dependenc
y':[3.2], 'status': 'N','score':7,'multi_task':[]},
          3.4:{'description':'Go hiking in Namsan Mountain -- Walk down t
he mountain and enjoy the sunset', 'duration':30, 'dependency':[3.3], 's
tatus': 'N','score':7,'multi_task':[]},
          4.1 :{'description':'Play table tennis in the res -- Find peopl
e who want to play too', 'duration':30, 'dependency':[], 'status': 'N',
'score':5,'multi_task':[1]},
          4.2 :{'description':'Play table tennis in the res -- Play a rou
nd with Gabrial', 'duration':15, 'dependency':[4.1], 'status': 'N','scor
e':5,'multi_task':[1]},
          4.3 :{'description':'Play table tennis in the res -- Play two r
ounds with Nikita who plays tennis', 'duration':15, 'dependency':[4.1],
'status': 'N','score':6,'multi_task':[1]},
          5.1 :{'description':'Get dinner with Mariia in haebang chon --
 Change to warm clothes', 'duration':5, 'dependency':[3.1, 3.2,3.3, 3.4,
4.1, 4.2,4.3], 'status': 'N','score':6,'multi_task':[]},
          5.2 :{'description':'Get dinner with Mariia in haebang chon --
 Climb upstairs', 'duration':10, 'dependency':[5.1], 'status': 'N','scor
e':4,'multi_task':[]},
          5.3 :{'description':'Get dinner with Mariia in haebang chon --
 Walk to the street full of grocery stores, street food, and restaurant
s', 'duration':10, 'dependency':[5.2], 'status': 'N','score':4,'multi_ta
sk':[]},
          5.4 :{'description':'Get dinner with Mariia in haebang chon --
 Wait for Korean orange chicken', 'duration':15, 'dependency':[5.3], 'st
atus': 'N','score':4,'multi_task':[]},
          5.5 :{'description':'Get dinner with Mariia in haebang chon --
 Take it back to the residential hall and eat together', 'duration':10,
'dependency':[5.4], 'status': 'N','score':3,'multi_task':[]},
          6 :{'description':'Discuss CS 11B assignment in my flatmates',
'duration':60, 'dependency':[], 'status': 'N','score':1,'multi_task':[1,
8.2]},
          7.1 :{'description':'Late night sight seeing with under Seoul t
ower -- Climb up in the woods while chatting with Yufei', 'duration':50,
'dependency':[5.1, 5.2,5.3,5.4], 'status': 'N','score':9,'multi_task':
[]},
          7.2 :{'description':'Late night sight seeing with under Seoul t
ower -- Reach the top, and appreciate the night view of Seoul', 'duratio
n':50, 'dependency':[7.1], 'status': 'N','score':9,'multi_task':[]},
          7.3 :{'description':'Late night sight seeing with under Seoul t
ower -- Go down', 'duration':50, 'dependency':[7.2], 'status': 'N','scor
e':9,'multi_task':[]},
          7.4 :{'description':'Late night sight seeing with under Seoul t
ower -- Buy Soju and chocolate', 'duration':50, 'dependency':[7.3], 'sta
```

```
tus': 'N','score':9,'multi_task':[]},
         8.1 :{'description':'Ready for bed  - Shower', 'duration':20,
'dependency':[7.4], 'status': 'N','score':5,'multi_task':[]},
         8.2 :{'description':'Ready for bed - Brush teeth', 'duration':5
, 'dependency':[8.1], 'status': 'N','score':5,'multi_task':[6]},
         8.3 :{'description':'Ready for bed - Say goodnight to roommate
 Dodie and rest in peace', 'duration':10, 'dependency':[8.2], 'status':
'N','score':8,'multi_task':[]},
        }
```

In [ ]:

```
#new input task printing, which includes multitasking
def print_input_new_tasks(tasks):
    """
    Input: list of tasks
    Task Status:
    - 'N' : Not yet in priority queue (default status)
    - 'I' : In priority queue
    - 'C' : Completed
    Output: print statement with all the tasks to be included in the sch
eduler
    """
    print('Input List of Tasks')
    for id, value in tasks.items():
        print(f"task:{id} \t {value['description']} \t duration:{value
['duration']} \t depends on: {value['dependency']} \t Status: {value['st
atus']} \t Happiness: {value['score']}, \t Multi_task: {value['multi_tas
k']}")

print_input_new_tasks(new_tasks)
```

```
Input List of Tasks
task:1   Go to CS 111B class     duration:90     depends on: []
Status: N        Happiness: 2,   Multi_task: [6, 4.1, 4.2, 4.3]
task:2   Go back to bed and take a nap   duration:60     depends on:
[1]      Status: N       Happiness: 8,   Multi_task: []
task:3.1         Go hiking in Namsan Mountain -- Wait Sam in the lobby
duration:10      depends on: [2]         Status: N       Happiness: 7,
Multi_task: []
task:3.2         Go hiking in Namsan Mountain -- Run all the staircases
in the mountain          duration:70     depends on: [3.1]       Statu
s: N     Happiness: 7,   Multi_task: []
task:3.3         Go hiking in Namsan Mountain -- Workout in the outdoor
gym on the top of the mountain   duration:40     depends on: [3.2]
Status: N        Happiness: 7,   Multi_task: []
task:3.4         Go hiking in Namsan Mountain -- Walk down the mountain
and enjoy the sunset     duration:30     depends on: [3.3]       Statu
s: N     Happiness: 7,   Multi_task: []
task:4.1         Play table tennis in the res -- Find people who want t
o play too       duration:30     depends on: []          Status: N
Happiness: 5,    Multi_task: [1]
task:4.2         Play table tennis in the res -- Play a round with Gabr
ial      duration:15     depends on: [4.1]       Status: N       Happin
ess: 5,          Multi_task: [1]
task:4.3         Play table tennis in the res -- Play two rounds with N
ikita who plays tennis   duration:15     depends on: [4.1]       Statu
s: N     Happiness: 6,   Multi_task: [1]
task:5.1         Get dinner with Mariia in haebang chon -- Change to wa
rm clothes       duration:5      depends on: [3.1, 3.2, 3.3, 3.4, 4.1,
4.2, 4.3]        Status: N       Happiness: 6,   Multi_task: []
task:5.2         Get dinner with Mariia in haebang chon -- Climb upstai
rs       duration:10     depends on: [5.1]       Status: N       Happin
ess: 4,          Multi_task: []
task:5.3         Get dinner with Mariia in haebang chon -- Walk to the
street full of grocery stores, street food, and restaurants      durati
on:10    depends on: [5.2]       Status: N       Happiness: 4,   Multi_
task: []
task:5.4         Get dinner with Mariia in haebang chon -- Wait for Kor
ean orange chicken       duration:15     depends on: [5.3]       Statu
s: N     Happiness: 4,   Multi_task: []
task:5.5         Get dinner with Mariia in haebang chon -- Take it back
to the residential hall and eat together         duration:10     depend
s on: [5.4]      Status: N       Happiness: 3,   Multi_task: []
task:6   Discuss CS 11B assignment in my flatmates       duration:60
depends on: []   Status: N       Happiness: 1,   Multi_task: [1, 8.2]
task:7.1         Late night sight seeing with under Seoul tower -- Clim
b up in the woods while chatting with Yufei      duration:50     depend
s on: [5.1, 5.2, 5.3, 5.4]       Status: N       Happiness: 9,   Multi_
task: []
task:7.2         Late night sight seeing with under Seoul tower -- Reac
h the top, and appreciate the night view of Seoul        duration:50
depends on: [7.1]        Status: N       Happiness: 9,   Multi_task: []
task:7.3         Late night sight seeing with under Seoul tower -- Go d
own      duration:50     depends on: [7.2]       Status: N       Happin
ess: 9,          Multi_task: []
task:7.4         Late night sight seeing with under Seoul tower -- Buy
Soju and chocolate       duration:50     depends on: [7.3]       Statu
s: N     Happiness: 9,   Multi_task: []
```

```
task:8.1          Ready for bed  - Shower          duration:20     depend
s on: [7.4]       Status: N       Happiness: 5,   Multi_task: []
task:8.2          Ready for bed - Brush teeth     duration:5      depend
s on: [8.1]       Status: N       Happiness: 5,   Multi_task: [6]
task:8.3          Ready for bed - Say goodnight to roommate Dodie and re
st in peace       duration:10     depends on: [8.2]       Status: N
Happiness: 8,     Multi_task: []
```

In [ ]:
```python
#Check if a task is already scheduled at a specific time
def fixed_task(ready_task, c_time):
  for task in ready_task:
    if (task[1]['id'] == 1) and c_time == 600: #if a task should be happ
en at this time

      #change its priority to infinity by converting tuple to list and c
onvert it back
      new_task = list(task)
      new_task[0] = float('-inf')
      new_task = tuple(new_task)

      #remove the task and give it a new score
      ready_task.remove(task)
      ready_task.append(new_task)

  return ready_task
```

The only fixed task here is CS 111B class, but creating a function allows me to add more if I have more fixed tasks in the future.

In [ ]:
```python
def completion_checker(score, id_dic, remain_time, pqueue):
  if remain_time == 0: #if we finish the task, we remove the dependency
      print('✅ Complete task:',id_dic['id'], new_tasks[id_dic['id']]['d
escription'],'at 🕒 ', int(c_time/60) ,':',int((c_time) % 60))
      remove_dependency(new_tasks, id_dic['id'])
  else: #if not, push the remaining task  back to task list
      pqueue.heappush((score, id_dic, {'duration':remain_time}))
```

I optimize my code by creating a completion checker, so I can reuse this code, when I am checking for multi-tasking too.

In [ ]:
```python
#Print out when multi-tasking
def Multi_task_printer(multi_id_dic,id_dic):
  global multi
  if (multi_id_dic,id_dic) not in multi:
    multi.append((multi_id_dic,id_dic))
    return True
  else:
    return False
```

In [ ]:
```
#Create a new function for multi-tasking because we will update the glob
al clock through the main task
def multi_task_time_update(task_time):
  """
    Implements time step calcluation for the scheduler
    Input: total task time
    Output: remaining task time after a time step
    """
  step = step_size #step is the scheduler's time step
  if task_time <= step: #If it is less than the step_size take a smaller
time step
    step = task_time
  task_time -= step # STEP 5: adjust the tasks remaining time
  return task_time
```

In [ ]:
```
def multi_task_checker(tasks,id_dic, pqueue):
  """
  Check if the current main task is multi-taskable, and find its multi-t
askable paired task.
  We pull out two candidate tasks to check.
  """
  muti_task_list = tasks[id_dic['id']]['multi_task']
  temporary_store = []
  if muti_task_list: #if we can multi_task
    if pqueue.heap:
      multi_score, multi_id_dic, multi_duration_dic = pqueue.heappop()
      non_muli_task = sub_multi_task(muti_task_list, multi_score, multi_
id_dic, multi_duration_dic, pqueue, new_tasks)

      if non_muli_task:
        temporary_store.append(non_muli_task)

        #pop the next next element
        if pqueue.heap:
          multi_score, multi_id_dic, multi_duration_dic = pqueue.heappop
()
          non_muli_task = sub_multi_task(muti_task_list, multi_score, mu
lti_id_dic, multi_duration_dic, pqueue, new_tasks)
          if non_muli_task:
            temporary_store.append(non_muli_task)

  for task in temporary_store:
    pqueue.heappush(task)
```

In [ ]:
```
def sub_multi_task(muti_task_list, multi_score, multi_id_dic, multi_dura
tion_dic, pqueue, tasks):
  '''
  Check if a candidate task can be multi-tasked with the main task.
  If yes, we implement and print out the multi-task.
  If no, we return the non muli-taskable candidate.
  '''

  if multi_id_dic['id'] in muti_task_list: #if the task is in in multi_t
ask list of the main task
    if Multi_task_printer(multi_id_dic,id_dic):#Print the multi_task
      print('😩 Multi-task:',multi_id_dic['id'], tasks[multi_id_dic['id'
]]['description'],'and', id_dic['id'],':', tasks[id_dic['id']]['descript
ion'], 'at', int(c_time/60) ,':',int((c_time) % 60))

    multi_remain_time = multi_task_time_update(multi_duration_dic['durat
ion']) #we execute this task together, and change its remaining time.
    completion_checker(multi_score, multi_id_dic, multi_remain_time, pqu
eue)  #check if this task is completed

  else: #if not multi-taskable, we store its id temporarily in another l
ist
    non_muli_task = (multi_score, multi_id_dic, multi_duration_dic)
    return non_muli_task
```

In [ ]:
```
# Inputs Parameters to the Scheduler
step_size = 5  # 5 mintues as a step size
c_time = 600  # current time is set to the initial time in minutes (10:0
0 AM = 10x60)
start = [] #a list to store tasks we have started
multi = [] #a list to store tasks we have multi-tasked

new_pqueue = MinHeapq()

#heapify the ready tasks into priority queue
def add_tasks_pqueue(ready_task):

  for key in ready_task:
      new_pqueue.heappush(key) #push the tasks into a heap format

  return new_pqueue.heap
```

In [ ]:

```
while unscheduled_tasks(new_tasks) or new_pqueue.heap: #if there are uns
cheduled tasks or ready tasks in the wait list
  #STEP 1: Extract  tasks ready to execute (those without dependencies)
  ready_task = get_ready_tsks(new_tasks)

  #STEP 2: Update the score for tasks which need to be executed at a fix
ed time
  updated_ready_task = fixed_task(ready_task, c_time)

  #STEP 3: Push the tasks onto the priority queue
  add_tasks_pqueue(updated_ready_task)

  if new_pqueue.heap:
    score, id_dic, duration_dic = new_pqueue.heappop()

    if Start_printer(id_dic):
      print('⭐ Start task:',id_dic['id'], tasks[id_dic['id']]['descript
ion'], 'at', int(c_time/60) ,':',int((c_time) % 60))

    #STEP 4: implement the multi task
    multi_task_checker(new_tasks,id_dic, new_pqueue)

    #STEP 5: implement the main task
    remain_time = task_time_update(duration_dic['duration'])

    #STEP 6: check if the main task is completed
    completion_checker(score, id_dic, remain_time, new_pqueue)
```

⭐ Start task: 1 Go to CS 111B class 10:00AM at 10 : 0
😩 Multi-task: 6 Discuss CS 11B assignment in my flatmates and 1 : Go to CS 111B class at 10 : 0
✅ Complete task: 6 Discuss CS 11B assignment in my flatmates at 🕒 10 : 55
😩 Multi-task: 4.1 Play table tennis in the res -- Find people who want to play too and 1 : Go to CS 111B class at 11 : 0
✅ Complete task: 4.1 Play table tennis in the res -- Find people who want to play too at 🕒 11 : 25
✅ Complete task: 1 Go to CS 111B class at 🕒 11 : 30
⭐ Start task: 4.2 Play table tennis in the res -- Play a round with Gabrial at 11 : 30
✅ Complete task: 4.2 Play table tennis in the res -- Play a round with Gabrial at 🕒 11 : 45
⭐ Start task: 4.3 Play table tennis in the res -- Play two rounds with Nikita who plays tennis at 11 : 45
✅ Complete task: 4.3 Play table tennis in the res -- Play two rounds with Nikita who plays tennis at 🕒 12 : 0
⭐ Start task: 2 Go back to bed and take a nap at 12 : 0
✅ Complete task: 2 Go back to bed and take a nap at 🕒 13 : 0
⭐ Start task: 3.1 Go hiking in Namsan Mountain -- Wait Sam in the lobby at 13 : 0
✅ Complete task: 3.1 Go hiking in Namsan Mountain -- Wait Sam in the lobby at 🕒 13 : 10
⭐ Start task: 3.2 Go hiking in Namsan Mountain -- Run all the staircases in the mountain at 13 : 10
✅ Complete task: 3.2 Go hiking in Namsan Mountain -- Run all the staircases in the mountain at 🕒 14 : 20
⭐ Start task: 3.3 Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain at 14 : 20
✅ Complete task: 3.3 Go hiking in Namsan Mountain -- Workout in the outdoor gym on the top of the mountain at 🕒 15 : 0
⭐ Start task: 3.4 Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset at 15 : 0
✅ Complete task: 3.4 Go hiking in Namsan Mountain -- Walk down the mountain and enjoy the sunset at 🕒 15 : 30
⭐ Start task: 5.1 Get dinner with Mariia in haebang chon -- Change to warm clothes at 15 : 30
✅ Complete task: 5.1 Get dinner with Mariia in haebang chon -- Change to warm clothes at 🕒 15 : 35
⭐ Start task: 5.2 Get dinner with Mariia in haebang chon -- Climb upstairs at 15 : 35
✅ Complete task: 5.2 Get dinner with Mariia in haebang chon -- Climb upstairs at 🕒 15 : 45
⭐ Start task: 5.3 Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants at 15 : 45
✅ Complete task: 5.3 Get dinner with Mariia in haebang chon -- Walk to the street full of grocery stores, street food, and restaurants at 🕒 15 : 55
⭐ Start task: 5.4 Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken at 15 : 55
✅ Complete task: 5.4 Get dinner with Mariia in haebang chon -- Wait for Korean orange chicken at 🕒 16 : 10
⭐ Start task: 5.5 Get dinner with Mariia in haebang chon -- Take it back to the residential hall and eat together at 16 : 10
✅ Complete task: 5.5 Get dinner with Mariia in haebang chon -- Take it

back to the residential hall and eat together at 🕒 16 : 20
⭐ Start task: 7.1 Late night sight seeing with under Seoul tower -- Climb up in the woods while chatting with Yufei at 16 : 20
✅ Complete task: 7.1 Late night sight seeing with under Seoul tower -- Climb up in the woods while chatting with Yufei at 🕒 17 : 10
⭐ Start task: 7.2 Late night sight seeing with under Seoul tower -- Reach the top, and appreciate the night view of Seoul at 17 : 10
✅ Complete task: 7.2 Late night sight seeing with under Seoul tower -- Reach the top, and appreciate the night view of Seoul at 🕒 18 : 0
⭐ Start task: 7.3 Late night sight seeing with under Seoul tower -- Go down at 18 : 0
✅ Complete task: 7.3 Late night sight seeing with under Seoul tower -- Go down at 🕒 18 : 50
⭐ Start task: 7.4 Late night sight seeing with under Seoul tower -- Buy Soju and chocolate at 18 : 50
✅ Complete task: 7.4 Late night sight seeing with under Seoul tower -- Buy Soju and chocolate at 🕒 19 : 40
⭐ Start task: 8.1 Ready for bed - Shower at 19 : 40
✅ Complete task: 8.1 Ready for bed - Shower at 🕒 20 : 0
⭐ Start task: 8.2 Ready for bed - Brush teeth at 20 : 0
✅ Complete task: 8.2 Ready for bed - Brush teeth at 🕒 20 : 5
⭐ Start task: 8.3 Ready for bed - Say goodnight to roommate Dodie and rest in peace at 20 : 5
✅ Complete task: 8.3 Ready for bed - Say goodnight to roommate Dodie and rest in peace at 🕒 20 : 15

Q5 [#ComputationalCritique]

A. Produce a critical analysis of your scheduler, highlighting:

all the benefits in following the algorithmic directives defined in the instructions:

- I like using a nested dictionary. Compared with class code, it provides better code readability to tell the value I'm extracting.
- I like I separate different jobs in functions, such as fixed_task and completion_checker. It allows me to recycle the code (completion_checker) and expand the code in the future (fixed_task).

and any failure modes and/or limitations you envision it running into:

- I don't like my switch from nested dictionary to tuple. I use it because I cannot heapify dictionary {'score': 1, 'id': 2, 'duration'} so I have to change it tuple and integer (1, 'id': 2, 'duration'). It causes me to debug time because I need to remind myself of the datatype.
- I don't like my Start_print and Multi_task_print because I need to create an extra list in global. My initial method is to change the task status. Still, it means that I need to push back a status variable that is inconvenient.
- I don't like multi-task checkers and sub-multi-task checkers because I feel they have too many input variables and are not independent enough. I cannot articulate what they are doing, respectively.

B. Examine the efficiency of your schedule (not scheduler), including making explicit reference to the metrics employed to determine this.

The scheduler shows that if I multi-task in my CS 111B class, I can finish my day and go to bed much earlier. I determine multi-tasking based on the relationship between two tasks and if the task is breakable. The advantage of using a multi-task list instead of True or False consider the relationship between two tasks. For instance, I can text while taking class, but I don't want to text while discussing with people. Another advantage of using the list is to avoid breaking some unbreakable tasks. For instance, I can text in class. However, I cannot stop having a class after I finish texting. Therefore, I put texting in my class's multi-task list, but I don't put a class in my texting's multi-task list to avoid conflict.

C. Will you start using your algorithm to schedule your day? Explain your answer in as much detail as possible. I won't use the scheduler because

1. I seldom multi-task. I prefer to focus on one thing at one moment. For instance, I never multi-task in CS110 class.
2. I consider more variables than the scheduler. For instance, I will think of my energy level and decide what to do. However, the energy level is unpredictable, so I won't be able to schedule it.
3. Accidents happen in my real life, such as some spontaneous trip. Human decision-making is much more complicated than what machines can predict.

HC Application

Breakitdown: I break down jobs into different functions to reuse my code for both my main task and muti-tasks. It helps me a lot when I try to debug because I know where to focus when getting an error. It also allows me to expand my code(e.g., set more fixed-scheduled tasks) without reading all the codes again.

Communicationdesign: My code communicates better with the audience compared with class code. First, I gather all the functions I need to initiate together (step_size, c_time, start, multi). It is easier to know which block we should run first to rerun the main scheduler. Second, I use start_printer and muti_task_print to print whenever the scheduler goes through the loop. Third, the start, complete, multi-tasking emoji also allows readers to understand the schedule without reading it carefully.

Audience: From data type, functions, and the main schedulers are all tailored to readers. For instance, I use class for heap, so readers will only see pop and push and don't need to worry about building a heap map in the main scheduler. Lastly, I choose to use dictionaries rather than lists to increase readability.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```