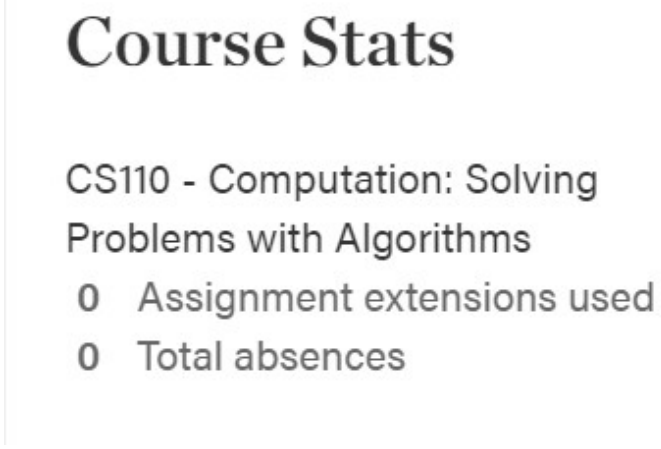
In [1]:
```
NAME = "Pedro Haschelevici"
COLLABORATORS = "Barbara Machado, Fellipe Couto, Gabriel da Silva"
```

CS110 Fall 2020 - Assignment 2

Question 0 [#responsibility]

In [2]:
```
from IPython.display import Image, clear_output
Image(filename="proof.jpg")
```

Out[2]:



Q1 [#ComputationalSolutions, #DataStructures]

In [3]:
```
Image(filename="tabela.jpg")
```

Out[3]:

	A	B	C	D	E	F	G
1	**id**	**description**	**sub-id**	**sub-tasks**	**duration (min)**	**depedencies**	**status**
2	**0**	**Morning routine**	0.1	Wake-up	5		n
3			0.2	Take a shower	25	0.1	n
4			0.3	Have breakfast	30	0.1	n
5	**1**	**Go to the Turkish Market**	1.1	Bike there	5	0.2, 0.3	n
6			1.2	Buy the ingredients	50	1.1	n
7			1.3	Bike back	5	1.2	n
8	**2**	**Cook the feast**	2.1	Mise en place	20	1.3	n
9			2.2	Cook the food	120	2.1	n
10			2.3	Organise the containers	10	2.2	n
11	**3**	**Picknick at the Park**			180	2.3	n
12	**4**	**Museum Island Tour**	4.1	Bike tour	30	3	n
13			4.2	Pergamon	75	3	n
14			4.3	Old museum	75	3	n
15	**5**	**Bike ride home**			15	3, 4.1, 4.2, 4.3	n
16	**6**	**Get Ready for the night**	6.1	Take a shower	25	5	n
17			6.2	Choose outfit	5	5	n
18			6.3	Have dinner	30	5	n
19	**7**	**A Beer with friends**			150	6.1, 6.2, 6.3	n

I chose to store my tasks in a dictionary of dictionaries, because I think it will be easier to access the values of the elements of the task.
I give all my tasks a priority score based on the numbers of dependencies it has, the lower the score, the higer the priority. After that, I use min heap to get the first tast.
To run the scheduler I increasse a timer for real time and one for the task and check if the task timer is runing for longer than the task, if yes, than the task is over. When it's over, I reset the task timer, pop the completed task, and hepify the priority queue to get the next task. Keep doing this until the priority queue is done

Q2 [#PythonProgramming, #CodeReadability]

In [4]:
```
#possible status
n = 'not done'
p = 'in progress'
c = 'completed'

tasks = {
         0.1: {'id':0.1, 'description':'Wake-up', 'duration_(min)':5,'dependencies': [],'status': n},
         0.2: {'id':0.2, 'description':'Take a shower', 'duration_(min)':25,'dependencies': [0.1],'status': n},
         0.3: {'id':0.3, 'description':'Have breakfast', 'duration_(min)':30,'dependencies': [0.1],'status': n},
         1.1: {'id':1.1, 'description':'Bike to turkish market', 'duration_(min)':5,'dependencies': [0.2,0.3],'status':
         1.2: {'id':1.2, 'description':'Buy the ingredients', 'duration_(min)':50,'dependencies': [1.1],'status': n},
         1.3: {'id':1.3, 'description':'Bike back', 'duration_(min)':5,'dependencies': [1.2],'status': n},
         2.1: {'id':2.1, 'description':'Mise en place', 'duration_(min)':20,'dependencies': [1.3],'status': n},
         2.2: {'id':2.2, 'description':'Cook the food', 'duration_(min)':120,'dependencies': [2.1],'status': n},
         2.3: {'id':2.3, 'description':'Organise the containers', 'duration_(min)':10,'dependencies': [2.2],'status': n
         3: {'id':3, 'description':'Picknick at the Park', 'duration_(min)':180,'dependencies': [2.3],'status': n},
         4.1: {'id':4.1, 'description':'Bike tour', 'duration_(min)':30,'dependencies': [3],'status': n},
         4.2: {'id':4.2, 'description':'Pergamon', 'duration_(min)':75,'dependencies': [3],'status': n},
         4.3: {'id':4.3, 'description':'Old museum', 'duration_(min)':75,'dependencies': [3],'status': n},
         5: {'id':5, 'description':'Bike ride home', 'duration_(min)':15,'dependencies': [4.1, 4.2, 4.3],'status': n},
         6.1: {'id':6.1, 'description':'Take a shower', 'duration_(min)':25,'dependencies': [5],'status': n},
         6.2: {'id':6.2, 'description':'Choose outfit', 'duration_(min)':5,'dependencies': [5],'status': n},
         6.3: {'id':6.3, 'description':'Have dinner', 'duration_(min)':30,'dependencies': [5],'status': n},
         7: {'id':7, 'description':'A Beer with friends', 'duration_(min)':150,'dependencies': [6.1,6.2,6.3],'status':
        }
```

In [5]:
```
def prio(d, l, score = 0):
    """
    Calculate the priority score of a task by the number of dependencies (nested)
    Input: dictionary of tasks, list of dependencies
    Output: priority score (int)
    """
    for i in l:  #iterate through the dependencies on the list
        score = prio(d, d[i]['dependencies'], score)+1  #check the dependencies of the dependencies
    return score

def addPrio(d):
    """
    Assign priority values using the prio function
    Input: dictionary of tasks
    Output: dictionary of tasks (with priority score)
    """
    for task in d:  #iterate through the tasks
        d[task]['score'] = prio(d, d[task]['dependencies'])    #get the priority scores
    return d

addPrio(tasks)
clear_output()
```

In [6]:

```
pq = list(tasks.values())   #priority queue

# Defining some basic binary tree functions
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 right child.

#create heapify function

def min_heapify(heap, i):
    """
    Input: heap : a list of floats
        Assume that the heap size is the length of the heap.

            i: index
    Note
    ----
    No output is needed. This function should modify (if necessary)
    heap in-place.

    """
    #index of both left child (l) and right child (r)
    #parent's index is i
    l = left(i)
    r = right(i)
    heapsize = len(heap)

    #check is there is a left child
    # if left child is smaller than parent, store l in "smallest"
    # if parent is still smaller than left child, store i in "smallest"
    if l < heapsize and heap[l]['score'] < heap[i]['score']:
        smallest = l
    else:
        smallest = i

    #check is there is a right child
    # if right child is smaller, store r in "smallest"
    if r < heapsize and heap[r]['score'] < heap[smallest]['score']:
        smallest = r

    # swap parent and child if the smallest value is a child
    # then recursivally calls heapify function
    if smallest != i:
        heap[i],heap[smallest] = heap[smallest],heap[i]
        min_heapify(heap, smallest)
    return(heap)

def my_min_heap(A):
    """
    heapify the whole piority queue
    Input: List of tasks
    Output: priority queue of tasks
    """
    for i in range(len(A)-1, -1, -1): #we use for loop to go through every element in the tree, starting form the botto
        min_heapify(A, i)
    return A
```

In [7]:

```
import time

def runSchedule(pq):
    """
    Runs the schedule
    Note: The duration of the tasks must be a multiple of 5 min
    Input: priority queue
    Output: No output
    """
    start = 480  #start time in minutes
    step = 10    #step size = 10 min

    t = start    #start the time
    task_time = 0    #start the task-specific time

    my_min_heap(pq)   #heapify pq
    while pq:   #while pq is not empty
        if task_time >= pq[0]['duration_(min)']:   #if task is completed
            if task_time > pq[0]['duration_(min)']:   #if some time passed after the task was completed
                t -= 5
            print(f"{t//60}:{t%60}")   #print the time (hr:min) and task status
            print(f"completed task: {pq[0]['description']}")
            pq[0]['status'] = c   #change the task status to completed
            pq.pop(0)   #pop the task from the list
            task_time = 0   #reset task time
            my_min_heap(pq)   #heapify pq
            continue

        if pq[0]['status'] == n:    #if the task has not started yet
            print(f"started task: {pq[0]['description']}")
            pq[0]['status'] = p     #change status to in progress

        elif pq[0]['status'] == p:   #if the task is in progress
            print(f"{t//60}:{t%60}")
            print (f"{pq[0]['duration_(min)'] - task_time} minutes till {pq[0]['description']} is done")

        t += step    #increasse time
        task_time += step
        time.sleep(1)   #wait 1 second
```

In [8]:
```
runSchedule(pq)
```

```
started task: Wake-up
8:5
completed task: Wake-up
started task: Take a shower
8:15
15 minutes till Take a shower is done
8:25
5 minutes till Take a shower is done
8:30
completed task: Take a shower
started task: Have breakfast
8:40
20 minutes till Have breakfast is done
8:50
10 minutes till Have breakfast is done
9:0
completed task: Have breakfast
started task: Bike to turkish market
9:5
completed task: Bike to turkish market
started task: Buy the ingredients
9:15
40 minutes till Buy the ingredients is done
9:25
30 minutes till Buy the ingredients is done
9:35
20 minutes till Buy the ingredients is done
9:45
10 minutes till Buy the ingredients is done
9:55
completed task: Buy the ingredients
started task: Bike back
10:0
completed task: Bike back
started task: Mise en place
10:10
10 minutes till Mise en place is done
10:20
completed task: Mise en place
started task: Cook the food
10:30
110 minutes till Cook the food is done
10:40
100 minutes till Cook the food is done
10:50
90 minutes till Cook the food is done
11:0
80 minutes till Cook the food is done
11:10
70 minutes till Cook the food is done
11:20
60 minutes till Cook the food is done
11:30
50 minutes till Cook the food is done
11:40
40 minutes till Cook the food is done
11:50
30 minutes till Cook the food is done
12:0
20 minutes till Cook the food is done
12:10
10 minutes till Cook the food is done
12:20
completed task: Cook the food
started task: Organise the containers
12:30
completed task: Organise the containers
started task: Picknick at the Park
12:40
170 minutes till Picknick at the Park is done
12:50
160 minutes till Picknick at the Park is done
13:0
150 minutes till Picknick at the Park is done
13:10
140 minutes till Picknick at the Park is done
13:20
130 minutes till Picknick at the Park is done
13:30
120 minutes till Picknick at the Park is done
13:40
110 minutes till Picknick at the Park is done
13:50
100 minutes till Picknick at the Park is done
14:0
90 minutes till Picknick at the Park is done
14:10
```

```
80 minutes till Picknick at the Park is done
14:20
70 minutes till Picknick at the Park is done
14:30
60 minutes till Picknick at the Park is done
14:40
50 minutes till Picknick at the Park is done
14:50
40 minutes till Picknick at the Park is done
15:0
30 minutes till Picknick at the Park is done
15:10
20 minutes till Picknick at the Park is done
15:20
10 minutes till Picknick at the Park is done
15:30
completed task: Picknick at the Park
started task: Bike tour
15:40
20 minutes till Bike tour is done
15:50
10 minutes till Bike tour is done
16:0
completed task: Bike tour
started task: Pergamon
16:10
65 minutes till Pergamon is done
16:20
55 minutes till Pergamon is done
16:30
45 minutes till Pergamon is done
16:40
35 minutes till Pergamon is done
16:50
25 minutes till Pergamon is done
17:0
15 minutes till Pergamon is done
17:10
5 minutes till Pergamon is done
17:15
completed task: Pergamon
started task: Old museum
17:25
65 minutes till Old museum is done
17:35
55 minutes till Old museum is done
17:45
45 minutes till Old museum is done
17:55
35 minutes till Old museum is done
18:5
25 minutes till Old museum is done
18:15
15 minutes till Old museum is done
18:25
5 minutes till Old museum is done
18:30
completed task: Old museum
started task: Bike ride home
18:40
5 minutes till Bike ride home is done
18:45
completed task: Bike ride home
started task: Take a shower
18:55
15 minutes till Take a shower is done
19:5
5 minutes till Take a shower is done
19:10
completed task: Take a shower
started task: Choose outfit
19:15
completed task: Choose outfit
started task: Have dinner
19:25
20 minutes till Have dinner is done
19:35
10 minutes till Have dinner is done
19:45
completed task: Have dinner
started task: A Beer with friends
19:55
140 minutes till A Beer with friends is done
20:5
130 minutes till A Beer with friends is done
20:15
120 minutes till A Beer with friends is done
20:25
110 minutes till A Beer with friends is done
20:35
```

```
100 minutes till A Beer with friends is done
20:45
90 minutes till A Beer with friends is done
20:55
80 minutes till A Beer with friends is done
21:5
70 minutes till A Beer with friends is done
21:15
60 minutes till A Beer with friends is done
21:25
50 minutes till A Beer with friends is done
21:35
40 minutes till A Beer with friends is done
21:45
30 minutes till A Beer with friends is done
21:55
20 minutes till A Beer with friends is done
22:5
10 minutes till A Beer with friends is done
22:15
completed task: A Beer with friends
```

Q3 [#ComputationalSolutions]

First change I made was to add a boolean value to check if the task is multi-taskable or not. Then I created a code that merges the first 2 tasks of the priority queue. Last, I check if the current task is multi-taskable or not, if it is, I check the next, if it is, I merge both of them, so they are executed at the same time.

Q4 [#PythonProgramming, #CodeReadability]

In [9]:

```
tasksM = {
         0.1: {'id':0.1, 'description':'Wake-up', 'duration_(min)':5,'dependencies': [],'multi_tasking':False,'status':
         0.2: {'id':0.2, 'description':'Take a shower', 'duration_(min)':25,'dependencies': [0.1],'multi_tasking':False
         0.3: {'id':0.3, 'description':'Have breakfast', 'duration_(min)':30,'dependencies': [0.1],'multi_tasking':True
         1.1: {'id':1.1, 'description':'Bike to turkish market', 'duration_(min)':5,'dependencies': [0.2,0.3],'multi_ta
         1.2: {'id':1.2, 'description':'Buy the ingredients', 'duration_(min)':50,'dependencies': [1.1],'multi_tasking'
         1.3: {'id':1.3, 'description':'Bike back', 'duration_(min)':5,'dependencies': [1.2],'multi_tasking':False,'sta
         2.1: {'id':2.1, 'description':'Mise en place', 'duration_(min)':20,'dependencies': [1.3],'multi_tasking':True,
         2.2: {'id':2.2, 'description':'Cook the food', 'duration_(min)':120,'dependencies': [2.1],'multi_tasking':True
         2.3: {'id':2.3, 'description':'Organise the containers', 'duration_(min)':10,'dependencies': [2.2],'multi_task
         3: {'id':3, 'description':'Picknick at the Park', 'duration_(min)':180,'dependencies': [2.3],'multi_tasking':T
         4.1: {'id':4.1, 'description':'Bike tour', 'duration_(min)':30,'dependencies': [3],'multi_tasking':False,'stat
         4.2: {'id':4.2, 'description':'Pergamon', 'duration_(min)':75,'dependencies': [3],'multi_tasking':False,'statu
         4.3: {'id':4.3, 'description':'Old museum', 'duration_(min)':75,'dependencies': [3],'multi_tasking':False,'sta
         5: {'id':5, 'description':'Bike ride home', 'duration_(min)':15,'dependencies': [4.1, 4.2, 4.3],'multi_tasking
         6.1: {'id':6.1, 'description':'Take a shower', 'duration_(min)':25,'dependencies': [5],'multi_tasking':False,'
         6.2: {'id':6.2, 'description':'Choose outfit', 'duration_(min)':5,'dependencies': [5],'multi_tasking':True,'st
         6.3: {'id':6.3, 'description':'Have dinner', 'duration_(min)':30,'dependencies': [5],'multi_tasking':True,'sta
         7: {'id':7, 'description':'A Beer with friends', 'duration_(min)':150,'dependencies': [6.1,6.2,6.3],'multi_tas
        }

addPrio(tasksM)
pqM = list(tasksM.values())   #priority queue
clear_output()
```

In [10]:

```
def mergeTasks(pq):
    """
    Merge the first 2 tasks
    Input: priority queue
    Output: No output
    """
    if pq[0]['duration_(min)'] >= pq[1]['duration_(min)']:    #if first is longer
        pq[0]['description'] += ' and ' + pq[1]['description']    #add second to firsts description
        pq.pop(1)   #pop second
    else:  #if second is longer
        pq[1]['description'] += ' and ' + pq[0]['description']    #add first to seconds description
        pq.pop(0)  #pop first

def runScheduleM(pq):
    """
    Runs the schedule
    Note: The duration of the tasks must be a multiple of 5 min
    Input: priority queue
    Output: No output
    """
    start = 480  #start time in minutes
    step = 10    #step size = 10 min

    t = start    #start the time
    task_time = 0    #start the task-specific time

    my_min_heap(pq)   #heapify pq
    while pq:   #while pq is not empty

        if len(pq)>1 and pq[0]['multi_tasking']:   #if the task is multi-taskable and there is a second task
            my_min_heap(pq[1::])    #get next task
            if pq[1]['multi_tasking']:     #if the next task is multi-taskable
                mergeTasks(pq)  #merge task with next task
                continue

        if task_time >= pq[0]['duration_(min)']:   #if task is completed
            if task_time > pq[0]['duration_(min)']:   #if some time passed after the task was completed
                t -= 5
            print(f"{t//60}:{t%60}")   #print the time (hr:min) and task status
            print(f"completed task: {pq[0]['description']}")
            pq[0]['status'] = c   #change the task status to completed
            pq.pop(0)   #pop the task from the list
            task_time = 0   #reset task time
            my_min_heap(pq)   #heapify pq
            continue

        if pq[0]['status'] == n:    #if the task has not started yet
            print(f"started task: {pq[0]['description']}")
            pq[0]['status'] = p     #change status to in progress

        elif pq[0]['status'] == p:   #if the task is in progress
            print(f"{t//60}:{t%60}")
            print (f"{pq[0]['duration_(min)'] - task_time} minutes till {pq[0]['description']} is done")

        t += step    #increasse time
        task_time += step
        time.sleep(1)   #wait 1 second
```

In [11]:
```
runScheduleM(pqM)
```

```
started task: Wake-up
8:5
completed task: Wake-up
started task: Take a shower
8:15
15 minutes till Take a shower is done
8:25
5 minutes till Take a shower is done
8:30
completed task: Take a shower
started task: Have breakfast
8:40
20 minutes till Have breakfast is done
8:50
10 minutes till Have breakfast is done
9:0
completed task: Have breakfast
started task: Bike to turkish market
9:5
completed task: Bike to turkish market
started task: Buy the ingredients
9:15
40 minutes till Buy the ingredients is done
9:25
30 minutes till Buy the ingredients is done
9:35
20 minutes till Buy the ingredients is done
9:45
10 minutes till Buy the ingredients is done
9:55
completed task: Buy the ingredients
started task: Bike back
10:0
completed task: Bike back
started task: Cook the food and Mise en place
10:10
110 minutes till Cook the food and Mise en place is done
10:20
100 minutes till Cook the food and Mise en place is done
10:30
90 minutes till Cook the food and Mise en place is done
10:40
80 minutes till Cook the food and Mise en place is done
10:50
70 minutes till Cook the food and Mise en place is done
11:0
60 minutes till Cook the food and Mise en place is done
11:10
50 minutes till Cook the food and Mise en place is done
11:20
40 minutes till Cook the food and Mise en place is done
11:30
30 minutes till Cook the food and Mise en place is done
11:40
20 minutes till Cook the food and Mise en place is done
11:50
10 minutes till Cook the food and Mise en place is done
12:0
completed task: Cook the food and Mise en place
started task: Organise the containers
12:10
completed task: Organise the containers
started task: Picknick at the Park
12:20
170 minutes till Picknick at the Park is done
12:30
160 minutes till Picknick at the Park is done
12:40
150 minutes till Picknick at the Park is done
12:50
140 minutes till Picknick at the Park is done
13:0
130 minutes till Picknick at the Park is done
13:10
120 minutes till Picknick at the Park is done
13:20
110 minutes till Picknick at the Park is done
13:30
100 minutes till Picknick at the Park is done
13:40
90 minutes till Picknick at the Park is done
13:50
80 minutes till Picknick at the Park is done
14:0
70 minutes till Picknick at the Park is done
14:10
60 minutes till Picknick at the Park is done
```

```
14:20
50 minutes till Picknick at the Park is done
14:30
40 minutes till Picknick at the Park is done
14:40
30 minutes till Picknick at the Park is done
14:50
20 minutes till Picknick at the Park is done
15:0
10 minutes till Picknick at the Park is done
15:10
completed task: Picknick at the Park
started task: Bike tour
15:20
20 minutes till Bike tour is done
15:30
10 minutes till Bike tour is done
15:40
completed task: Bike tour
started task: Pergamon
15:50
65 minutes till Pergamon is done
16:0
55 minutes till Pergamon is done
16:10
45 minutes till Pergamon is done
16:20
35 minutes till Pergamon is done
16:30
25 minutes till Pergamon is done
16:40
15 minutes till Pergamon is done
16:50
5 minutes till Pergamon is done
16:55
completed task: Pergamon
started task: Old museum
17:5
65 minutes till Old museum is done
17:15
55 minutes till Old museum is done
17:25
45 minutes till Old museum is done
17:35
35 minutes till Old museum is done
17:45
25 minutes till Old museum is done
17:55
15 minutes till Old museum is done
18:5
5 minutes till Old museum is done
18:10
completed task: Old museum
started task: Bike ride home
18:20
5 minutes till Bike ride home is done
18:25
completed task: Bike ride home
started task: Take a shower
18:35
15 minutes till Take a shower is done
18:45
5 minutes till Take a shower is done
18:50
completed task: Take a shower
started task: Have dinner and Choose outfit
19:0
20 minutes till Have dinner and Choose outfit is done
19:10
10 minutes till Have dinner and Choose outfit is done
19:20
completed task: Have dinner and Choose outfit
started task: A Beer with friends
19:30
140 minutes till A Beer with friends is done
19:40
130 minutes till A Beer with friends is done
19:50
120 minutes till A Beer with friends is done
20:0
110 minutes till A Beer with friends is done
20:10
100 minutes till A Beer with friends is done
20:20
90 minutes till A Beer with friends is done
20:30
80 minutes till A Beer with friends is done
20:40
70 minutes till A Beer with friends is done
20:50
```

```
60 minutes till A Beer with friends is done
21:0
50 minutes till A Beer with friends is done
21:10
40 minutes till A Beer with friends is done
21:20
30 minutes till A Beer with friends is done
21:30
20 minutes till A Beer with friends is done
21:40
10 minutes till A Beer with friends is done
21:50
completed task: A Beer with friends
```

Q5 [#ComputationalCritique]

I think overall, my code works pretty well. The use of a dictionary, in the beginning, makes it easier for it to get a specific task by id. Also, the organization of the sub-tasks with sub-ids makes more clear for someone to intuitively understand the id of the sub-task. The sorting by dependencies adds more realism to the schedule, as it mimics the process that people usually do to prioritize tasks.

One thing that I do not appreciate in this code is the time-step because it limits the code from processing the real timing of events. For next time I might try a more continuous approach, maybe reducing the step-size to the limit as it approaches 0. Another thing that I might do differently next time is adding an importance level for tasks, so if I have more tasks that I can do in 24 hours then I prioritize some over others.

This algorithm is not a very efficient way to do this scheduling, both computationally and in a practical way. In real life, I probably would not use this, since it takes too much effort to plug-in the tasks, and I would probably be able to do my schedule in my head with no problems (like I've been doing my whole life). Computationally speaking, this algorithm uses a very big number of steps, since it calls the heapify function various times for each task done. However, even this not being very computationally efficient, we don't need to care a lot about that due to the always small input size. A day has only a limited amount of hours, so you can only do a limited amount of tasks, this constrains the input-size (number of tasks) to always be small, making the code easy to run even in the slowest modern computers. Even if we plugged in 20 tasks a day for a whole month, it would only be n = 600, which for a time complexity analysis is a very small number.

HCs

#algorithms - Throughout the whole assignment I used this HC to either create my codes, analyze codes or understand how a code worked in order to come up with my code.

#organization - I used this HC on my codes, with the use of docstrings and comments, and also the general format of the code, I made it more organized and easier for both myself and other people to understand what is it doing.

#critique - I used critique on question 5 to criticize my own code and understand its strengths and weaknesses, and how could I improve it for the next time.

In [ ]: