Reflection

Question 1:
When we design the inputs, we firstly think about what approach will be used to solve the problem. We guess that the solver will use a loop to find the answer from the case of all students in one room or from the case of each student in a room. So, we assume the optimal room number is equal to ½ x (one room + maximum number of rooms) as N. Then we divide students into N rooms and students who are divided into the same room will share a higher mutual happiness. Then adjust the stress between them in order to meet the requirement. In addition, happiness and stress between students who are in different rooms are close to that between students in the same room. Otherwise, it is easy to separate students by clustering students. Furthermore, I assume the solver will use the greedy algorithm, some local optimization is designed. To be precise, the students with the highest happiness will be put into two rooms. And they two also share a very high stress. They also have very low happiness and high stress with each other's roommates. In this case, if the solver chooses these two students in one room, other students cannot be put into the same room with them and total happiness will be very low. The same strategy also is applied to the students with lowest stress.

If we have more time, we will design more pitfalls for different algorithms, like local search, max weight clique.

Question 2:
We utilized the Greedy algorithm and considered different scenarios. We think it's a good approach as it leads to better results each step. Also, it does not take too much time to process as we are certain that it would stop after max(number of student) iterations. We considered four different scenarios. First, we start with a single room. This is like a top down approach where we assign all students in one room. Then we calculate the happiness produced by each student. Then we assign the one with the smallest produced happiness to another room. Each step we choose the room with the most stress and select one student in the room who produces smallest happiness. Then we iterate through all other rooms and put the student in the first room which does not break the stress constraint. We stop the algorithm whenever all rooms at the point meet the stress constraint. The second scenario is pretty similar to the first scenario. The only difference is that we select the student who produces the most stress in the room and assign the student to another room. By starting with 1 room, we start with the potentially largest happiness, and we reduce the happiness until stress constraint is met, then the happiness produced is still large.

Question 3:
We come up with the third and fourth scenarios at first. Which is more like a bottom up approach rather than a top down approach. For the third scenario, at first, we assigned all the students to a separate room. We then sort all the edges of the graph according to the value of happiness, selecting the largest edge and merging the two rooms where the two endpoints belong to. If the merge does not meet the stress constraint, we repeat the process for the second highest happiness edge. If we cannot merge any of the two rooms after traversing all the edges, the algorithm terminates. The fourth scenario is similar to the third scenario, except that we sort all the edges by the value of stress, selecting the smallest edge and merging the two rooms where the two endpoints belong to. This approach works well for inputs that emphasize stress constraints where you have to put students in many rooms. But it does not perform as well as the top down greedy algorithm if the number of rooms needed is not large.

We also tried a brutal force approach where we iterate through all possible results. The result is guaranteed to be optimal. However, this approach takes exponential time. So we abandon this approach and just use it to do some experiment to test the optimality of our greedy solution.

Question 4:
We did not use any fancy computational resources in this project. We only used our own laptops to do all the calculations. We write our functions and generate results in Jupyter Notebook.

Question 5:
For this project, we have tried two main ideas, namely greedy algorithms and brute force. If given more time, we may want to try to reduce this problem to other NP problems(such as Clique). In addition, we may also want to use a randomization algorithm and restart: each time we can pick a random initial solution(randomly assign students to rooms which meet the stress constraint) and rerun the greedy algorithm to finally get a better solution.