1.

a. An optimal substructure, by definition, is, if the problem in question, can be broken down into sub-problems to find an optimal solution to its subproblems, whilst, efficently. The optimal solution in this problem is to complete all the steps in the lowest number of switches. The optimal substructure by virtue of being done by the same student and their ability to have a consecutive scheduled list, we can select a plan with minimum switches in n steps

b. A greedy algorithm that can be applied here is to schedule those students with the greatest amount of consecutive steps. If we list in non-descending order, we'll have a method to minimize the instances of switches between the students that, of course, fulfill the requirements, that does the basics of an greedy algorithm: make the best instant/local decision in hopes of a globally optimized solution. Pick students with the longest consecutive-scheduled number of jobs that have the first job scheduled. After so, look for the longest consecutive jobs.

c. Coded

d. My algorithm at the beginning starts with a while loop. "0 < steps..." Once, steps has decremented to 0 then it will stop. At the worst case, my algorithm should take O(n^3), The code relating to student that finish the greatest number of steps. Taking into account both instances, the worst case is O(n^3).

e. Assertion: Greedy Algorithm (GAB) described in B, returns an optimal solution.

Proof by contradiction:

Assume that there exists an optimal solution (OPT) that is a better solution than the greedy algorithm (GAB) that I proposed. OPT can produce a list of scheduling times for the given students with less switches relative to GAB and the same number of experiments as GAB.

Let the switches that GAB produces be:

$GAB = \{ G_1, G_2, G_3, \ldots, G_T \}$ where T is arbitrary