Problem 1

(a) Describe the optimal substructure of this problem

Ans- The optimal substructure of this problem will be to schedule the student who signed up for the most consecutive steps, go first. And the subproblem(s) would be to schedule next student with most consecutive steps that do not overlap with previous student.

(b) Describe the greedy algorithm that could find an optimal way to schedule the students

Ans- Find all the students that can do the first step and pick the student that can do the most consecutive steps. Repeat this process until all the steps are taken.

(d) What is the runtime complexity of your greedy algorithm? Again, you don't need to factor in the setup of the lookup table, just your scheduling algorithm.

Ans- The runtime complexity of this problem would be $O(n^2m)$. n=steps and m=students.

(e) In your PDF, based on your answer to part b, give a full proof that your greedy algorithm returns an optimal solution.

Ans-Suppose we have my algorithm ALG and optimal algorithm OPT.
The solutions from ALG are a1, a2, a3, ……..aj and k switches
The solutions from OPT are o1, o2, o3….oj and l switches.
l<k
Lets assume that until the i-th iteration both algorithms are the same.
Using cut and paste method we get a1, a2,…..ai and oi+1, oi+2….oj.
Both algortihms are set up to pick the student with most consecutive steps so replacing ai and oi will give the same number of switches.
This is a contradiction to our previous claim of l<k.
Hence why ALG is optimal.

Problem 2

(a) Describe an algorithm solution to this problem. Feel free to talk about how you would adapt an algorithm we covered in class.

Ans-The algorithm that can be implemented for this problem is Dijkstra's algorithm for the shortest path between the stations. It will also compute the wait time for when a train is not available at the current time. Since we are calculating the time it makes from one station to another, we start by adding X=5:30 to get the start time and then select the appropriate vertex that represent station S(Starting point). We use the algorithm to compare the cost(minutes) from u to v for each pair of vertices, we also have to take into consideration for the wait time based on arrival time and frequency at v to add that to the cost to get the overall cost at each pair of vertices(u,v).

(b) What is the complexity of your proposed solution in (a)?

Ans-The time complexity for the solution in (a) is $O(V^2)$

(c) See the file FastestRoutePublicTransit.java, the method "shortestTime". Note you can run the file and it'll output the solution from that method. Which algorithm is this implementing?

Ans- The algorithm the method "shortestTIme" implements is Dijkstra's shortest path algorithm.

(d) In the file FastestRoutePublicTransit.java, how would you use the existing code to help you implement your algorithm? The existing code only handles one piece of data per edge, so describe some modifications.

Ans-We need to check for the arrival time and frequencies of the trains in each station. So we need to modify that part of the code. This will help modify the wait time on each stations and help with comparisons in the shortest path algorithm. We can add another array to keep track of the frequencies and times.

(e) What's the current complexity of "shortestTime" given V vertices and E edges? How would you make the "shortestTime" implementation faster? Describe any algorithm changes or data structure changes. What's the complexity of the optimal implementation?

And- The current complexity of “shortestTime” is $O(V^2)$. We can make it faster by performing algorithm on binary heap, which will give us the O(E log V0 complexity. And If we decide on doing it on fibonnaci heap, the complexity can be O(E + V log V).