A:
1. What were your results from compare_cow_transport_algorithms? Which algorithm runs faster? Why?
The greedy algorithm runs faster because it is searching a smaller total amount of possible trips. Instead of calculating every single combination of cows and choosing the best one, like the brute force algorithm, it just calculates the number of trips if you always take the cows heaviest to lightest. Thus, once it has found a single set of trips that uses all the cows, it stops, instead of calculating all other possible trips.

2. Does the greedy algorithm return the optimal solution? Why/why not?
No. because it just looks at the heaviest cow that can fit, it will exclude certain optimal cow arrangements, such as if 2 lighter cows and 1 heavy cow fill up more of a spaceship than just 2 heavy cows.

3. Does the brute force algorithm return the optimal solution? Why/why not?
Yes. because the brute force algorithm finds every possible solution, it must thus find the optimal solution. Then it is as simple as searching for said optimal solution.

B:
1. Explain why it would be difficult to use a brute force algorithm to solve this problem if there were 30 different egg weights. You do not need to implement a brute force algorithm in order to answer this.
With 30 different egg weights, there are something like 30! (2.7e32) different possible combinations. For a brute for algorithm to calculate every single one of these combinations and then pick the best valid would take, for all practical purposes, infinite time.

2. If you were to implement a greedy algorithm for finding the minimum number of eggs needed, what would the objective function be? What would the constraints be? What strategy would your greedy algorithm follow to pick which ~~coins~~ (did they mean eggs?) to take? You do not need to implement a greedy algorithm in order to answer this.
The objective function would be to find the combination of eggs which fully fills the ship (aka the sum of which = target weight) without going over and has the fewest total eggs used. The constraints would be that the sum of the egg weights cannot exceed target weight. The greedy algorithm would function by first trying to add the egg of the highest weight. It would keep doing this until adding a new egg of that weight would make the total weight go over target weight. At that point, it would try to add the egg of the next highest weight. So on until the egg weights sum to target weight.

3. Will a greedy algorithm always return the optimal solution to this problem? Explain why it is optimal or give an example of when it will not return the optimal solution. Again, you do not need to implement a greedy algorithm in order to answer this.

Example of a non optimal solution:
Target_weight = 100
Egg_weights = (1,50,51)
Optimal:(50,50)
Greedy:(51,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1)