1)

Devrim Arabaları is a film that narrates an attempt to start serial domestic production of cars in Turkey, in 1961. President of the day, who started his administration with a coup in 1960, asked the manufacturing of a car from engineers of TCDD (Turkish National Railways). Due date was the 38th birthday of the Turkish Republic, 29 October 1961. President Cemal Gürsel wanted it to be demonstrated to public during celebrations. Nearly thirty engineers worked hard to finish the project on time. They didn't have much time, it was only 130 days away from the Republic Day when they were assigned with it. It had to be designed and produced from scratch. So, they first study the already existing cars that the individuals have in team by dismembering each part. Although the time is limited, they managed to build three cars. They transported the cars from the factory in Eskişehir to Ankara where the celebrations will be held. President Cemal Gürsel visited Anıtkabir and completed the ceremonial runs with Devrim. But this is only the happy parts and happy ending of the story, not the all of it. From start to end engineers faced with a lot of difficulty. Not only about the manufacturing the car but also, they had so much pressure on them due to the people's attitude towards them. Many people were seeing them as a bunch of group who just waste the money of government. It is always said that we (Turkish People) are not open for new things. This story and the film based on that story shows that thought might be true. People always said say things like "You can't do it, there is no way, it is impossible, you are wasting your time". But this is not true. There is no way you know what will or can be happen if you work, work hard to accomplish something that you really want to accomplish. The impossible is just about time, in my opinion. Things that seem impossible to accomplish or happen at the moment. But this does not mean they will remain impossible until the end of time. It is just about trying. Waiting there, not stepping out the bad and common thoughts will not get anyone to anywhere. I hope, not just our people but humanity will understand it someday.

2)

Algorithm is simple due to brute force method. Hard part is finding each possible answer array. Once we get the arrays (which we get from permutation) remaining job is just finding a total hour for each array. The array which gives minimum hours of work is our true answer. So, return it after the loop ends. I implemented Heap's algorithm for permutation* as suggested on moodle. There is no different complexity cases for this algorithm since it computes every single possibility. Heap's algorithm is a recursive algorithm which does an addition to an array for base case and a swap on other cases. So creating each permutation takes $\Theta(N)$ time which evaluates $N!$ time for each permutation. Taking summation of hours as a constant job, this algorithm would work in $\Theta(N!)$ time

```
procedure generate(n : integer, A : array of any):
    if n = 1 then
          output(A)
    else
        for i := 0; i < n - 1; i += 1 do
            generate(n - 1, A)
            if n is even then
                swap(A[i], A[n-1])
            else
                swap(A[0], A[n-1])
            end if
        end for
        generate(n - 1, A)
    end if
```

Algorithm:

1. The algorithm generates (n-1)! permutations of the first n-1 elements, adjoining the last element to each of these. This will generate all of the permutations that end with the last element.
2. If n is odd, swap the first and last element and if n is even, then swap the i^{th} element (i is the counter starting from 0) and the last element and repeat the above algorithm till i is less than n.
3. In each iteration, the algorithm will produce all the permutations that end with the current last element.

Algorithms taken from:

http://www.geeksforgeeks.org/heaps-algorithm-for-generating-permutations/

https://en.wikipedia.org/wiki/Heap%27s_algorithm

3)

The basic need is the number of connected graphs in given map. Once found, problem can be solved by placing a lab to each graph and building n-1 (n represents vertex number) roads. Total cost would be lab count (which is equal to connected graph count) times lab cost plus (total vertex count – connected graph count) times.

4)

List to sort: [12, 34, 54, 2, 3]

Insertion Sort	
12, 34, 54, 2, 3	2, 12, 34, 3, 54
12, 34, 54, 2, 3	2, 12, 3, 34, 54
12, 34, 54, 2, 3	2, 3, 12, 34, 54
12, 34, 54, 2, 3	
12, 34, 54, 2, 3	
12, 34, 2, 54, 3	Shell Sort
12, 2, 34, 54, 3	12, 34, 54, 2, 3
2, 12, 34, 54, 3	12, 34, 54, 2, 3

12, 2, 54, 34, 3

12, 2, 3, 34, 54

3, 2, 12, 34, 54

2, 3, 12, 34, 54

Color Coding:

Green Numbers: Seems sorted for that step

Red Numbers: Are currently compared with each other

Black Numbers: They aren't considered yet

Shell sort is an algorithm similar to the bubble/insertion sort but with multiple gaps that are higher than one. This gives an advantage to sorting nearly sorted lists or to the ones that have lesser elements (or bigger elements if sorting in reverse order) on the end of the list. Because carrying an item from the end to the beginning of the list requires n times of comparison and swaps. So basically, shell sort uses bubble/insertion sort like algorithm but with bigger steps to speed up.