Problem 1: (15 points)

You manage the kids’ room at a busy airport. This room has got a small cot on which only one baby can sleep at a time. Since babies are of different ages and they are flying from different time-zones, their sleep pattern (when does a child want to sleep and for how long) is totally arbitrary.

The parents hand in the request for a time-slot for sleeping in the format (sleep-start-time, sleep-end-time). Only one request for time slot (of any size) is allowed per child and your job is to refuse the minimum number of parents.

You use the following Greedy strategy. On a time-scale, first you draw all the sleep-requests. Then you choose the baby whose requested sleep-period overlaps with least number of other babies (ties broken arbitrarily). Now, if this baby is given the cot, obviously the other kids whose sleep-period overlaps with this baby cannot be accommodated, so eliminate them from your consideration. Repeat the process until no more kid can be accommodated.

Does this strategy necessarily result in minimum number of refusals? If yes, prove it. Else, just provide a counterexample.

Problem 2: (20 points)

Consider the following algorithm for computing the lengths of the Single Source Shortest Paths (positive, integral weight) for all vertices in an $n$-vertex graph (say from vertex 1):

1. // dist[] is initialized to all infinities, except dist [1] = 0.
2. // edgeCost[i,j] = cost of the edge between i & j, or infinity if no edge exists
3. // see choice #2 for the extra line of code to be inserted here
4. 
5. for $i = 1 .. n$
6.     for $j = 1 .. n$
7.         dist [$i$] = min { dist [$i$] , dist [$j$] + edgeCost[$i$,$j$] }

Choose and justify exactly one of the following choices:

1) This algorithm works fine for all the cases. Compare the complexity of this algorithm with the Dijkstra's algorithm. If you think it is it more efficient than Dijkstra’s, then justify the reason why it is not preferred over Dijkstra's algorithm for computing shortest paths.

2) It works for some cases but not for others. Explain in which cases it will work (give example) and cases where it will not work (give counterexample). If we run the piece of code
above $n$ times (which can be implemented by adding one more for $k = 1 \ldots n$ loop at line #5), will it make the algorithm foolproof? Justify your claim.

3) It does not work for any case. Justify why it will not work for any case. Just showing a single counterexample is not enough since that does not cover all the cases.

Problem 3: (15 points)

You are riding a spaceship and you need to go from star $A$ to star $B$. These stars are part of a huge galaxy consisting of $n$ stars. Space travel is not entirely safe, and you are given a chart $P[i][j]$ (of $n \times n$ size), where $P[i][j]$ represents the probability of safe journey of a direct route (without stopping at any other star in between) from star $i$ to star $j$, and $0 \leq P[i][j] \leq 1$ for all $(i,j)$ pair. You can assume that $P(i,j)$ is same as $P(j,i)$. All the probability figures are independent of each other, so the overall probability of safe journey of a path $(A \rightarrow x \rightarrow y \rightarrow \ldots \rightarrow z \rightarrow B)$ = $P(A, x) \times P(x, y) \times \ldots \times P(z, B)$. Design an algorithm to calculate the highest probability of safe journey between $A$ and $B$.

Problem 4: (15 points)

There are various clubs at UF (e.g., drama club, tennis club, swimming club). Each student is a member of one or more clubs based on their talents (each students has at least one talent). You have the membership records of all the clubs, and each record (array) is sorted on UF-ID.

Your goal is to determine a minimal set of clubs (called $T$) so that their membership covers each and every student at UF. You start with a null club-list ($T$) and a null student-list ($L$).

Your algorithm works this way. First you add to $T$ the club with highest membership (say Football club). So, add the member of the Football club to $L$. Next, you see which club has the highest membership among the students who are yet to make it to your list $L$. Add that club to $T$ and add the members of that club to $L$. Keep on going this way until all the students in UF are on your list $L$.

Does the above Greedy algorithm always yield an optimal solution? If yes, give a formal proof why it does. Otherwise, provide a counterexample.

Problem 5: (15 points)

You want to chop a long piece of wooden stick into smaller pieces of different but pre-defined lengths. You achieve this with marking the stick where the cuts are to be made. For chopping a $L$-length stick into two pieces, the carpenter charges $L$ dollars (does not matter whether the two pieces are of equal length or not, i.e, the chopping cost is independent of where the chopping point is but depends on the length of the piece to be cut). Therefore, the order of the choosing the chopping points is important: if you instruct the carpenter to start chopping as per the cut marks
from one end of the stick to the other, this may not result in the minimum cost (see the example below).

Consider a greedy approach to solve this problem:

Choose a chopping point nearest to the center of a stick since that leaves two pieces that are about the same length. Recursively apply this algorithm to each of the subparts till no more cuts have been made. Does this Greedy algorithm necessarily yield an optimal solution? If yes, prove it. Else, give a counterexample where it does not work.

**Example:** take an 8” stick with cut-marks at every inch. A bad strategy would be to chop it at the 1” mark, then 2” mark, then 3” mark … which will cost 8+7+6+ … + 2 = $35. However, if you choose the Greedy strategy above, the cutting points would be at 4”, (2”/ 6”), (1”/ 3”/ 5”/ 7”), which would cost 8 + (4+4) + (2+2+2+2) = $24.

![Bad strategy](image1.png)

![Greedy strategy](image2.png)

Remember, the cut marks could have been ANYWHERE, not just at all the 1” spaces.

**Problem 6: (20 points)**

You are responsible for restoring the telephone connectivity between two cities after a hurricane damaged the exiting telephone network. A telephone network consists of cables and switches (switches interconnect two or more cables). The prices of the cables vary by their length, while the price of the switch varies by their quality and load-sharing ability. You have the map of the whole network, which tells you what are the position & quality requirement of the switches (which means their different prices), and the length of the cable between two directly connected switches (which again translates to their prices). You do not want to revamp the whole network – you just want to restore a single connection that would connect the two cities at the minimum price.

Reduce this problem into a graph problem on which Dijkstra’s Single Source Shortest Path algorithm may be applied. Describe how you will do the transformation and give the complexity of your transformation algorithm.