***1) Consider a mobile company that wants to connect wireless clients to base stations. Further suppose that each base station has a capacity of C clients. A client can connect to a base station only if the base station set is less than or equal to distance r away. Given a collection of base stations, wireless clients and the geographic location of base stations and wireless clients, design a method to determine an assignment of clients to base stations such that the number of clients serviced is maximized.***

The problem stated is very clear, a variation of the ***Maximum Coverage Problem***, which is a well-studied optimization problem in Computer Science. The goal is to maximize the number of clients that can be assigned to a base station while ensuring some restrictions are met.

Firstly, there is an objective function to be maximized: ***“number of clients serviced”.***

Secondly, apart from maximizing the objective function, some constraints as follows must be respected:

1. ***Each base station has a capacity of C clients.***
2. ***Clients can connect to a base station if the set-distance is less than or equal to distance “r” away.***

Once having clearly stated the problem, in order to solve this problem, different approaches can be used. For instance, a linear programming approach can be solved by any optimization method as it could be the well-known ***max-flow algorithm***, as we have seen in lectures.

The wireless clients will be the source nodes, and the base stations will be the sinks to which we should arrive. Therefore, given that situation, the maximum capacity of the edges connecting the clients to the base stations will be the solution to the question of the maximum capacity of each base station.

This problem can be addressed using a graph-based assignment algorithm, such as the ***Maximum Matching of a graph***. First, construct a bipartite network, where base stations are one set of nodes and clients are the other set of nodes. Next, connect a base station and a client if the distance between the base station and the client is less than or equal to ***“r”***. Finally, find a maximum weight correspondence in the bipartite network, where the weight of a matching edge is the capacity of the base station. In this way, the maximum number of clients will be served and the capacity constraints of each base station will be satisfied.

For example, we just have to solve the problem by running a ***max-flow algorithm*** in order to, for example, know how many clients (maximum) can be connected properly to the base stations. It is important to double check that each client connected to any base station is not further away than the distance ***“r”*** to the base station. In case of finding this situation, we should remove the link between the customer and the base station due to the fact that the second requirement would not be met.

***Greedy algorithm***

The most common approach to solving this problem is to use a greedy algorithm. The algorithm starts by selecting the base station with the most available capacity and assigns as many clients as possible to that base station. The process is then repeated for the remaining base stations and clients. This algorithm can be efficient and yields near-optimal results for many cases.

***Integer Programming***

Another approach is to use ***Integer Programming***, which can provide an exact solution. However, this approach is more computationally expensive and may not scale well for large numbers of clients and base stations.

In summary, the ***Maximum Coverage Problem*** can be solved using ***maximum matching of a graph***, ***greedy algorithms*** or ***integer programming***, depending on the requirements for accuracy and computational efficiency.

***2) There are “n” doctors that need to be scheduled for duty in a hospital during holiday periods. There are m vacation periods (Thanksgiving, Christmas etc.) each of period pi . However there must be at least one doctor on call at the hospital every day during the vacation period. Each doctor Di specifies the set of vacation days Wi that he can work. How will you determine a schedule such that each doctor is not working more than c days in any vacation period.***

***n*** = number of doctors

***m*** = number of vacation periods

***PI*** = length of the vacation

***Di*** = each doctor’s set of vacation days that can work

***Wi*** = vacation days

***Constraint*** = at least one doctor on call every vacation day

***“c”*** = maximum cap of vacation days working in any vacation period

The problem in question is the well-known ***Resource Constrained Project (RCP)*** scheduling problem. The objective is to schedule resources such that each employee does respect the restrictions as follows:

1. ***Ensuring there is always at least one employee on call every day during the vacation.***
2. ***Not working more than “c” days in any vacation period.***

***Integer Linear Programming***

The most usual approach to solving this problem is the ***Integer Linear Programming (ILP).*** In this specific context, a binary variable is assigned to each doctor for each day, representing whether the physician is scheduled to work or not.

The ***ILP*** model should take into account the constraints of each doctor's available vacation days and the maximum number of days a doctor can work. We have not to forget that, the ultimate goal of the ***ILP*** model is to maximize the number of days the hospital is fully staffed while respecting the constraints.

***Greedy Algorithm***

Another approach is to use a ***greedy algorithm***, which starts by assigning the doctor’s with the most available vacation days to the period with the fewest available doctor’s (always taking into account the ***“c”*** days restriction. The process is repeated for the remaining doctor’s and periods. This algorithm can be efficient, but does not necessarily provide an optimal solution as sometimes happens with the ***Greedy approach***.

Both approaches can be used, but a trade-off between accuracy and optimal solution, vs computational speed will be encountered. Thus, resulting in the ***ILP*** method the optimal one, but the most expensive one in terms of computational cost and time.

In order to give a more detailed step by step solution, the ***ILP*** method will be further explained .

1. ***Formulate the problem:***
   1. For each doctor, create a binary variable xi,j to indicate whether doctor i works on day j.
   2. For each vacation period, create a constraint to ensure that there is at least one doctor working each day.
   3. For each doctor, create a constraint to ensure that he/she does not work more than c days.
2. ***Solve the ILP:*** The objective function of the ***ILP*** is to minimize the number of doctors working. The constraints ensure that the number of doctors working is minimal and that each doctor does not work more than ***“c”*** days.
3. ***Extract the solution:*** The solution to the ***ILP*** gives the assignment of physicians to vacation days in such a way as to minimize the number of physicians working while ensuring that each physician does not work more than ***“c”*** days.

* This ***ILP*** formulation can be solved using a solver such as the simplex method.
* A max-flow approach could also be a suitable solution to the problem by using the theory learnt in class, but after reading in Internet the author has considered that the ***ILP*** method combined with a SIMPLEX solver would be the most-optimal solution also applying new concepts (SIMPLEX) that we will get exposure to in the next weeks of the Combinatorial Optimization course

***3) Describe the proof of Hall’s theorem discussed in class for a bipartite graph G = (U, V, E). In particular, in the induction step show that if there is a subset S ⊆ U with neighbor set N(S), where |N(S)| = |S| then S and T satisfy the hypothesis of the theorem.***

***i.e. for every subset in S( or T) the neighborhood set is of size greater than or equal to S (or T).***

***Hall's theorem*** is a result of graph theory that provides a necessary and sufficient condition for a bipartite graph to have perfect matching. The theorem states that a bipartite graph ***G = (U, V, E)*** has perfect matching if and only if, for every subset ***S*** of vertices in ***U***, the size of the neighborhood of ***S*** is at least as large as the size of ***S***.

The proof of ***Hall's theorem*** is usually done by induction. In the induction step, we assume that the theorem holds for all smaller subsets of ***U*** and show that it holds for ***S***. If ***|N(S)| = |S|,*** then we can pair each vertex of ***S*** with a distinct vertex of ***N(S).*** This pairing provides a one-to-one correspondence between ***S*** and ***N(S).***

Since the pairing is one-to-one, it follows that for each subset ***T*** of ***S***, the size of the neighborhood of ***T***, ***N(T)***, is at least as large as the size of ***T***. Therefore, the induction step follows and ***Hall's theorem*** is proved.

In summary, if there exists a subset ***S ⊆ U*** with a neighbor set ***N(S),*** where ***|N(S)| = |S| ,*** then the induction step of ***Hall's theorem*** shows that ***S*** and ***T*** satisfy the hypothesis of the theorem.

***4) Show that y(θ) = θx+ (1−θ)y, 0 ≤ θ ≤ 1 represents all the points on the line segment defined by two endpoints x and y. This is true in Rd for any dimension d.***

Regarding the given statement, our approach will be to assess the function in the boundaries of the variable θ. From 0 to 1 values. The equation ***y(θ) = θx + (1-θ)y*** defines a line segment in any dimension between the points x and y. For ***θ = 0***, ***y(θ) = (1-0)y = y***. For ***θ = 1***, ***y(θ) = θx + (1-1)y = θx + (0)y = x***. Therefore, ***y(θ)*** interpolates between ***y*** and ***x*** as ***θ*** varies from ***0*** to ***1***, passing through all points on the line segment defined by ***x*** and ***y*** on any dimension, not only the subspace of a plane, like a straight line.

***5) Show that the following sets are convex:***

***General definition of convex:***

A set S is convex if for every pair of points x, y in S, the line segment joining x and y is also in S.

***(i) Lines and Line segments in Rd .***

A line segment is convex because it can be defined as the convex combination of its endpoints. A line is also considered convex because it is the limit of the line segments as their length approaches infinity. Both of them fulfill the condition that *for every pair of points x, y in S, the line segment joining x and y is also in S.*

***(ii) Ellipsoids in Rd .***

An ellipsoid is defined as the set of points x on Rd such that (x-c)^T A^(-1) (x-c) <= 1 for some positive definite matrix A and vector c. It is a convex set because for any pair of points x, y on the ellipsoid, the convex combination θx + (1-θ)y is also on the ellipsoid since it satisfies the inequality:

(θx + (1-θ)y - c)^T A^(-1) (θx + (1-θ)y - c) = θ^2 (x-c)^T A^(-1) (x-c) + (1-θ)^2 (y-c)^T A^(-1) (y-c) + 2θ(1-θ)(x-c)^T A^(-1) (y-c) <= θ^2 + (1-θ)^2 = 1.

Therefore, as the condition *“A set S is convex if for every pair of points x, y in S, the line segment joining x and y is also in S”* is met, the Ellipsoids in Rd are convex.

***(iii) Halfspaces defined by the hyperplane a^T x = b.***

A half-space is defined as the set of points x in Rd satisfying the inequality a T x <= b for some vector a and scalar b. It is a convex set because for any pair of points x, y in the half-space, the convex combination θx + (1-θ)y is also in the half-space:

a T (θx + (1-θ)y) = θ a T x + (1-θ) a T y <= θ b + (1-θ) b = b.

Therefore, the convex property is met and the subset is convex.

***6) Let C ⊆ Rn be a convex set, with x1 . . . xk ∈ C. Show that θ1x1 + . . . + θkxk ∈ C where θ1, . . . θk ∈ R satisfy θi ≥ 0, θ1 + · · · + θk = 1.***

By definition, a set ***C*** is convex if for any two points ***x*** and ***y*** in ***C***, the line segment connecting ***x*** and ***y*** is also in ***C***. Since the line segment connecting ***x*** and ***y*** can be parameterized by a convex combination of ***x*** and ***y***, it follows that for any θ ∈ [0, 1], ***θx + (1-θ)y*** ∈ C.

Therefore, it can be shown that ***θ1x1 + θ2x2 + ... + θkxk*** ∈ C by induction. To prove this, we can take the case ***k=2*** and extend the result to ***k*** larger by induction. If ***k=2***, then ***θ1x1 + θ2x2 = θ1x1 + (1-θ1)x2*** ∈ C, which follows from the definition of convex set.

In general, if the result holds for ***k-1***, then ***θ1x1 + θ2x2 + .... + θkxk = θ1x1 + (1-θ1)(θ2x2 +...+ θkxk)*** ∈ C, since ***θ2x2 + ... + θkxk*** is a convex combination of ***x2, ..., xk*** and ***C*** is convex.

Therefore, it can be concluded that ***θ1x1 + θ2x2 + ... + θkxk*** ∈ C for any non-negative ***θ1, θ2, ..., θk*** that satisfies ***θ1 + θ2 + ... + θk = 1***.

***7) Show that if C is a convex set then C’ = T(C) is a convex set, where T(C) = Ax + b, x ∈ C .***

If C is a convex set, then its image ***T(C) = Ax + b***, where ***A*** is an **m x n** matrix and ***b*** is a vector in ***Rm***, is also a convex set.

To prove this, let us consider two points ***x1*** and ***x2*** in ***C***, and let ***θ ∈ [0, 1].*** Then

***T(θx1 + (1-θ)x2) = A(θx1 + (1-θ)x2) + b = θAx1 + (1-θ)Ax2 + b***

Since ***A*** is a linear transformation, it follows that ***θAx1 + (1-θ)Ax2*** is also a linear combination of ***Ax1*** and ***Ax2.***

Since ***C*** is convex, it follows that ***θx1 + (1-θ)x2*** is in ***C***, and therefore ***T(θx1 + (1-θ)x2)*** is in ***T(C).*** Therefore, ***T(C)*** is a convex set since it contains all convex combinations of its points.