Question 1

Consider a polytope of the form $\mathcal{P} = \{x \in \mathbb{R}^m : Ax \leq b\}$. We want to generate n uniformly random points from a polytope given by $\mathcal{P}$ such that no two points are within distance d of each other. In order to draw such a sample $X = (x_1, \cdots, x_n)$, we do the following :

- Initialize $X = (x_1, \cdots, x_n)$ such that $||x_i - x_j|| \geq d \ \forall i \neq j$
- Set iterations = 0
- While iterations $\leq$ max iterations (pre-specified)
  - Select a random component of $X$and call it i.
  - Generate a uniformly random point from $\mathcal{P}$ and call it y
  - If $||y - x_j|| \geq d \ \forall j \neq i$
    - $x_i = y$
    - iteration = itertion + 1

1. Take $d = 0.001$

$$A = \begin{bmatrix} 1 & -2 \\ -1 & -1 \\ 0 & -1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ -8 \\ -4 \\ 6 \\ 11 \end{bmatrix}$$

2. (**R**) What are the extreme points of this polytope. Plot the lines and use your judgement to come up with the extreme points. Do not try to come up with an algorithm in full generalization.
3. Find a rectangle parallel to axes which perfectly covers/encloses the given polytope. Now how will you draw points randomly from this given polytope? (Hint: rejection sampling)
4. (**R**) Take the initial sample to be n appropriately spaced points along any edge of the described polytope. Generate $n = 20$ points from the polytope $\mathcal{P} = \{x \in \mathbb{R}^m : Ax \leq b\}$ such that no two points are within distance d of each other.
5. (**R**) Plot the points for different values of max iterations = 30, 50, 100 ,200. Comment on the plots.
6. (**R**) Ignore the d separation condition and use rejection sampling to draw samples from $\mathcal{P}$. Estimate the area/volume of $\mathcal{P}$.
7. (**R**) Try to write a general function which will find all the extreme points for two dimensional case with inputs $A \in \mathbb{R}^{n \times 2}$ and $b \in \mathbb{R}^n$

Question 2 (Simulated Annealing)

We will now use simulated annealing which is a probabilistic technique for approximating the global optimum of a given function. Our aim is to solve a Travelling Salesman Problem (TSP) using this technique.

1. (**R**) Describe travelling salesman problem.

2. We will consider a small instance of the problem with 11 cities. The distance between any two cities are given in *TSP11.csv*.
   Try numpy.loadtxt(open("filename", "rb"), delimiter=",") to read the csv file.

3. The following is the pseudo-code for simulated annealing. We start with a initial random state. A state is given by a particular sequence of cities which the salesman travels. This is known as a "tour" of the salesman. Note that the salesman should travel all the cities only once.

Algorithm 1 Simulated annealing

```
1: Generate a random initial solution and set $T_0$
2: Calculate its cost using cost() function
3: For $k = 0$ through $max_{iter}$,
    a: Generate a random neighbouring state of the current solution using neighbour() function and calculate it's cost.
    b: $T_k$ = updated temperature given by cooling schedule.
    c: If $c_{new} \leq c_{old}$, move to new solution, else move to new solution with some probability given by accept_prob() function
    d: In each iteration store the best solution you got till that time
4: Return the best solution.
```

4. (**R**) Given a tour of the salesman compute the cost, which is total distance travelled in that tour. Create *cost()* function which takes a sequence of cities as input and calculates the total distance travelled. For example in a 4 city scenario, given the sequence of cities as [B,C,D,A], your function should calculate the total distance travelled in that particular order.

5. (**R**) Now we will create the *neighbour()* function. In TSP, neighbour of a state is defined as the new states obtained by swapping two consecutive cities. For example, in 4 city case, every state has 4 neighbours and neighbour of [A,B,C,D] are [B,A,C,D], [A,C,B,D], [A,B,D,C], [D,B,C,A]. Your function should choose one of the neighbours uniformly.

6. The acceptance probability is given by $e^{-(c'-c)/T_k}$ where c' and c are the cost of new state and the old state. T_k is the current temperature of the k^{th} iterate. Explain the behaviour of acceptance probability when (1) new cost gets more worse than the current one and (2) temperature decreases.

7. Usually, the temperature is started at high value and slowly decreased to 0 in each iteration by a cooling schedule. Consider the cooling schedule given by $T_{k+1} = \alpha T_k$ for $\alpha \geq 0.80$ and $T_0 = 1$.

8. (**R**) Run the simulated annealing algorithm and report the solution. Plot the cost function for each iteration and explain. Choose different T_0, α values and observe the effect.

9. (**R**) Come up with another cooling schedule (not of the form $T_{k+1} = \alpha T_k$) and comment on the change in behaviour of the algorithm.

10. Do the same for the slightly larger problem with 48 cities. The distance matrix is given in *TSP48.csv*