E.g. if your name is ABC, roll number is 2020csx1234 and submission is for lab2 then you should name the zip file as: ABC_2020csx1234_lab2.zip

PART I

Gibbs Sampler [Total 20 points]

Assume we are interested in simulating from the bivariate Gaussian distribution $N_2(\mu, \Sigma)$ where

$$\boldsymbol{\mu} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & a \\ a & 1 \end{pmatrix}$$

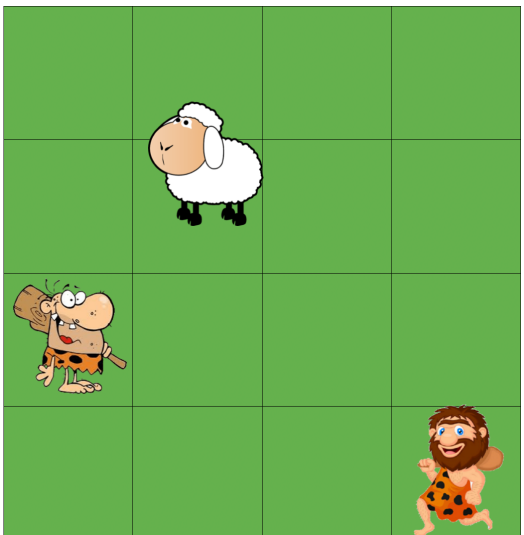
A. Derive the conditional distributions for x1 given x2 (i.e. p(x1/x2)) and x2 given x1 (i.e. p(x2/x1)). [5 **points**]
B. Implement the Gibbs sampler based on the conditional distributions. [5 **points**]
C. Run the algorithm for 10,000 iterations for a = 0. Use simulations from the standard Gaussian distribution as starting points. Estimate μ and Σ from your simulations. Use traceplots to see how many of the first samples you should discard. Make a plot of your simulations in the two-dimensional space plotting both x1 & x2, drawing a line between each iteration. Also plot the samples taken directly from the bivariate Gaussian distribution $N_2(\mu, \Sigma)$. Comment on the results. [5 **points**]
D. Now repeat the previous sampling with a = 0.99. Make similar estimates and plots and comment on the results. [5 **points**]

PART II

Prey and Predator model using Bayesian Network

(Total 20 points)

In this problem, two cavemen from Stone-age are trying to capture a sheep. Obviously the sheep can run faster than the cavemen. The cavemen are starving and if they do not get the sheep within a certain amount of time (turns) they may die out of hunger.



The strategy to capture the sheep involves predicting where the sheep is going to flee to or try to lure it into a corner.

The rules for the game can be seen in the list below:

1. The environment is a board/matrix with each tile defining a position
2. As mentioned, there are two types of players on the board (cavemen & sheep)
3. At most one player can be positioned on each tile
4. When a caveman is placed on a tile next to the sheep, the sheep is captured by the caveman and the game ends.
5. The sheep can move at most two tiles at a time, but only diagonally with both steps in the same direction.
6. The cavemen can move one tile at the time in only vertical and horizontal directions.
7. The sheep cannot be captured diagonally by the caveman
8. A player can choose not to move during a turn
9. All players make their move at the same time

Note: Consider board size is of 8*8 for reasonable run time during implementation.

Below are the four parts of your assignment that need to be completed. Design these scenarios using matrix and observe the different outputs.

[Part 1: 5 marks] – A sheep that works with a simple deterministic algorithm, that tries to evade the cavemen and avoid getting trapped in a corner. – Two cavemen that work with a simple deterministic algorithm that always tries to follow and capture the sheep – The board has no obstacles. – All positions of other players are available to all others. Play 2 games, upto 100 movements and output the agent that wins.

[Part 2: 15 marks] – One of the cavemen (caveman 1) uses a Bayesian Network (see Fig.1.) that tries to predict the sheep's next move and thereby *learn the sheep's strategy (cavemen otherwise have no access to the sheep's algorithm of movement in the grid)*. Implement the Bayesian caveman using the information in the Help below. Test the system by playing 5 games, upto 200 movements and output the agent that wins.

Help--
Caveman's Simple Algorithm

```
For each possible move
  Calculate the distance to sheep
Select shortest distance to sheep after move, and execute.
```

Sheep's Algorithm

```
IF there is no cavemen within radius (2 moves of a caveman)
       do not move
       ELSE for each possible move
        calculate the sum of: distance to nearest corner,
        distance to caveman 1 and distance to caveman 2,
          for each possible move.
            Choose the move with the highest sum.
```

Implementation of the game

The main idea is that we have a list of cavemen, each with an ID, and a list of sheep who also have an ID. These are used to perform the placement of the players on the board, and once this is done each of the players are asked to perform a choice as to where they want to move. Once all players have given their answer the players are moved to their target, and board configuration is tested for whether it is a winning configuration or not. If the game is not won the players are once again asked to perform an action, and so on.

Note: If the game has not been won in 200 turns the game should be terminated (as the game might have entered a loop).

Bayesian Network Setup

For coupling the Bayesian Network together with the caveman agent in the code, a set of input and output is required. The input for the Bayesian Network is given by the agent code, and the Bayesian Network returns the new probabilities. These are used to make an output choice.

The agent gets this information by analyzing the game board. It will use the following Input/Output:

Input : Location of Sheep, Location of fellow Caveman, Own Location
Output : Move to choose (North, South, East, West, Stay)

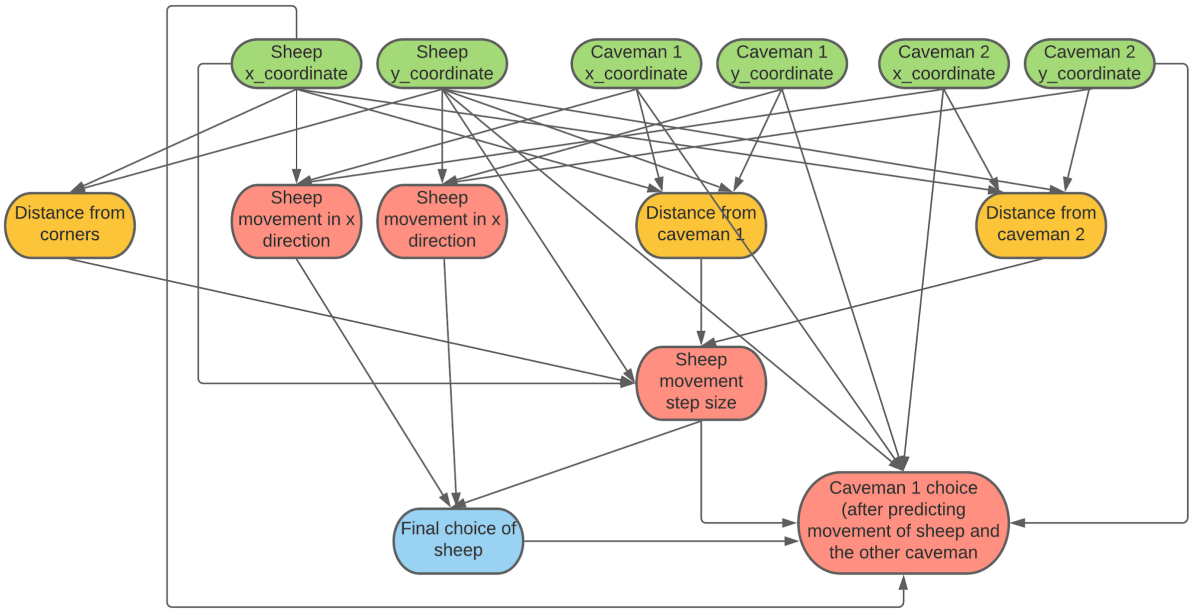


Fig.1.: Combined Caveman and Sheep model. Caveman 1 assumes that caveman 2 uses the min-euclidean distance method to predict the sheep. This means that if we know his location, the sheep's location and our guess of what choice the sheep makes we should be able to learn sheep's strategy from this. The green variables are observable, orange and blue variables are scripted and red variables are to be learned.

Sheep's Choice Decision

Expression for Choice, which combines DirectionX, DirectionY and Step into the final choice of the sheep.

```
if(Step==”Stay”)
        Stay;
if(Step==1)
        if(DirectionX==”East”)
                if(DirectionY==”North”)
                        Move North-East
                else
                        Move South-East
        else
                if(DirectionY==”North”)
                        Move North-West
                else
                        Move South-West
```

```
else
    if(DirectionX==”East”)
        if(DirectionY==”North”)
            Move 2 units North-East
        else
            Move 2 units South-East
    else
        if(DirectionY==”North”)
            Move 2 units North-West
        else
            Move 2 units South-West
```

Training

Run about 100000 iterations of the game using a random start location for each actor in each game until the network is trained sufficiently (the inferences in the bayesian network is stable).

Testing

Play 5 games upto 200 movements and output the agent that wins. See if Caveman 1 wins more.

Tie Braking

In case if more both the cavemen try to move to the same tile, the first cavemen in the list would be allowed to make the move. The order of choice making is the two cavemen, first and second, and then the sheep.

Lookahead

If the cavemen are to utilize their Bayesian Networks, that now are capable of predicting the movement of the other actors in the game, a lookahead is needed. The idea behind the lookahead is that you use the prediction of the other actors for each possible move you may choose. If you find a winning situation within one of these possible moves you choose it, or else you try again and make predictions of your possible moves after your first series of possible moves. This can be done as many times as required.

Note: To keep memory constraints in check, a lookahead of 1 step is used.

We decided to let the caveman make a random move 20% of the time. This would allow situations where the caveman causes the sheep to make a choice it would not have taken.