


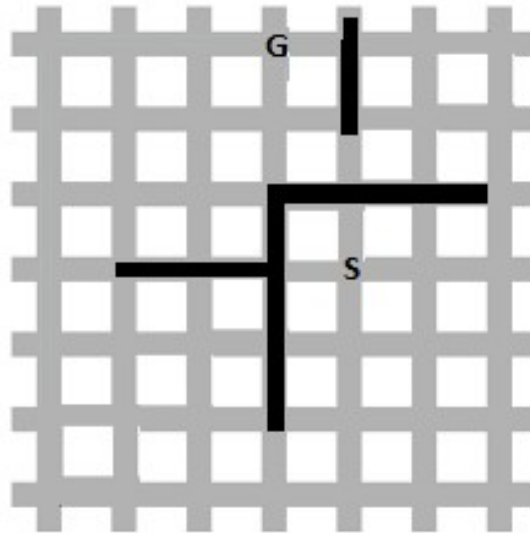
## National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Artificial Intelligence	Course Code:	AI 2002
	Program:	BS (Data Science)	Semester:	Spring 2024
	Duration:		Total Marks:	
	Paper Date:		Weight	4%
	Section:	A, B	Page(s):	
	Exam Type:	Final Assignment		

### Question [Search Algorithms]

[5 + 4 + 5 + 6 Points]

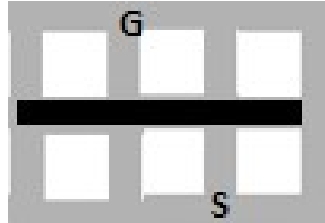
A company needs to write a program for an autonomous vehicle to plan a path from a starting point **S** to a destination **G** in a block town (a sample is shown below).



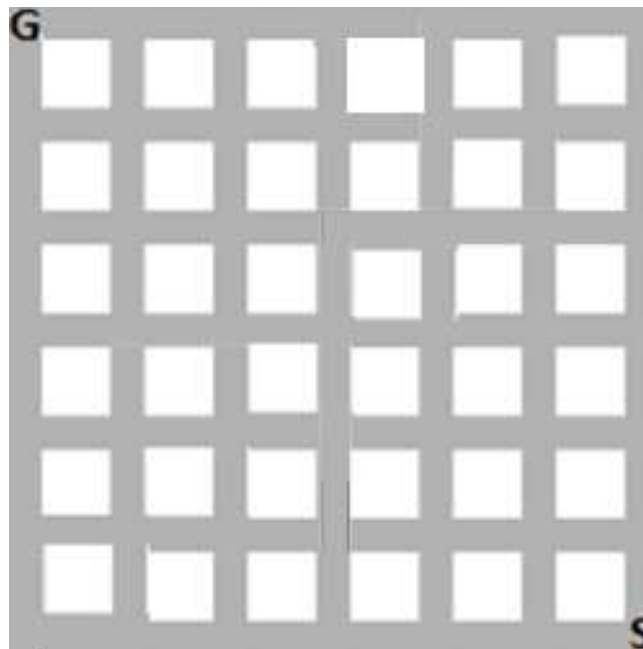
Each crossing in the town is connected to at most 4 other crossings. The CEO of the company (most probably a FAST-LHR alumni) decided to use A\* algorithm for planning the path. Some roads in the town are closed for maintenance and are marked using **BLACK** color. Further, the distance between neighboring crossings is not same so some neighboring crossings are close whereas some are far from each other (The above picture does not capture this situation). For simplicity you can assume that the state of the entire road network is completely visible at all times and that we only need to plan paths between various crossings of the town. Answer the following questions for this problem.

- How would you represent state of the vehicle during the path planning?
- Describe at least two different heuristic functions that can be used to guide the A\* search algorithms.

- c) For a simple town shown below show complete working of A\* algorithm to find a path from S to G. You might assume that all crossings are equidistant from the neighboring crossings. You must show the state of Queue after the expansion of each node that comes out of queue during the algorithm execution.



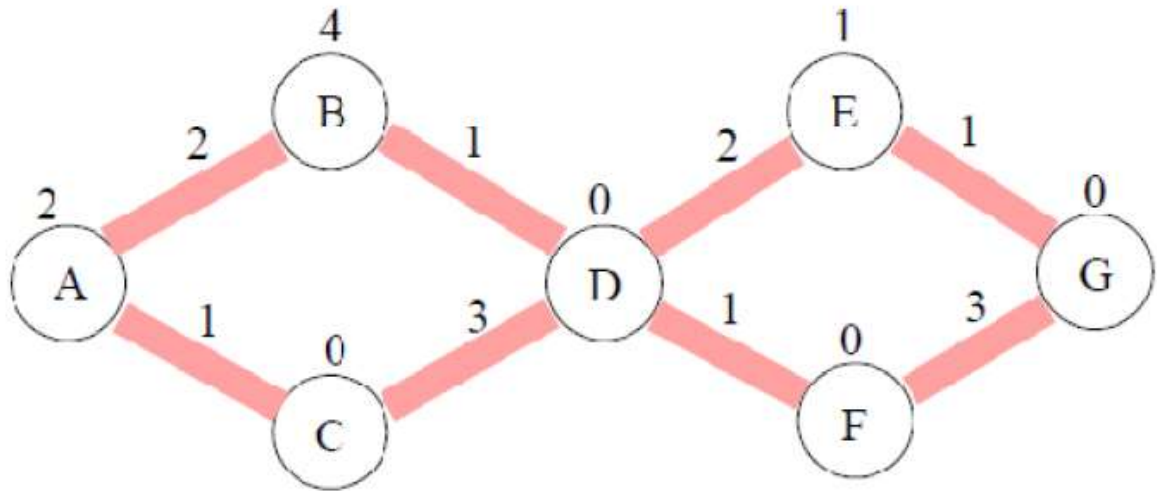
- d) For the following starting state compute an estimate of minimum number of nodes expanded by A\*, BFS, DFS.



Question [Search Algorithm Again]

[10 Points]

The following is a state-space search graph of a problem with nodes labeled using alphabets, edges are the thick shaded lines. The number above each edge giving the transition cost (e.g.,  $\text{cost}(C,D) = 3$ ) of the corresponding action and the number above each node is its heuristic value of that node (e.g.,  $h(A) = 2$ ). The node labeled A is the initial state and that labeled as G is the goal state.



- List, in order, the nodes that will be expanded if A\* algorithm is used to find a path from the initial state to the goal state. **[2 Points]**
- What path (if any) would be found by A\* algorithm in part a. **[2 Point]**
- What would be the state (nodes in it) of frontier/queue when the goal is found by A\* algorithm in part b. **[1 Point]**
- In one of my implementation of A\* I made an interesting mistake. My implementation was identical to the correct A\*, except that when it visits a node  $n$  that has already been expanded, it immediately skips  $n$  instead of checking if it needs to reinsert  $n$  into the priority queue. What path would be found by my erroneous implementation of A\* in the search problem given in part a above. Also comment on the effects of this change on the optimality of A\* algorithm. **[3 Points]**
- One FAST- student suggested an early termination strategy for A\* algorithm. He suggested that A\* must be stopped as soon as the first goal node G is found in some successor list instead of waiting until G is popped off the priority queue. What path would be found by his version of A\* for the state space search problem

given in part a. Also comment on the effects of this change on completeness and optimality of A\* algorithm. **[3 Points]**

**Question [Local Search]:**

An intelligent but greedy automated agent/program is trying to solve the famous 0-1 knapsack problem. In this problem the agent has a set of items to choose from with each item having a weight the agent has to carry if it picks that item, and a value the agent will gain by picking up the item. The agent has the capacity to carry only a **MAX\_WEIGHT** while his job is to select a subset of items that it can carry such that the total value gained by the agent is by picking up the subset of items is maximum.

The agent has been programmed by a selected team of students from a famous University (FAST) situated in the city of Lahore. The team programmed the agent such that it uses hill-climbing strategy to solve such optimization problems.

To represent state of the agent, while it is creating a plan to make a selection from a set of  $n$  items, the team used an array of size  $n$  with an index corresponding to one of the item in the set. Each index in the state array contains either a 0 or a 1 where 0 at any index means that the agent will not pick the corresponding item and a 1 means the agent will pick the corresponding item.

Further, to generate successors of a state a very simple successor function is used that works by selecting an index and if the corresponding item is not already in the list of items to be picked it added it into the list by placing a 1 at the corresponding array index.

Part a) If the state of items contains 100 items [1 + 1 + 3 Points]

- i. What is the size of search space the agent has to search from? (Justify)
- ii. How many successors a state can have at max? (Justify)
- iii. What will be a suitable evaluation function to carry out the hill-climbing search?

Part b) Now suppose that the set of items contain only 4 items. The weights of items and their value is given in the following table.

Item ID	A	B	C	D
Item Weight	1 kg	1 kg	2 kg	3 kg
Item Value	Rs. 100	Rs. 100	Rs. 300	Rs. 300

Table 1: Item Weights and Value

If the maximum capacity of the agent is 4 kg. and it starts in the following state then what will be final solution found be the agent (Show Complete Working) [5 Points]

Initial State	0	0	0	0
---------------	---	---	---	---

**Question [Perceptron and Perceptron Learning] [4 + 6 Points]**

Figure 1 show a single perceptron with two inputs X1 and X2 and a bias term that is permanently set to.

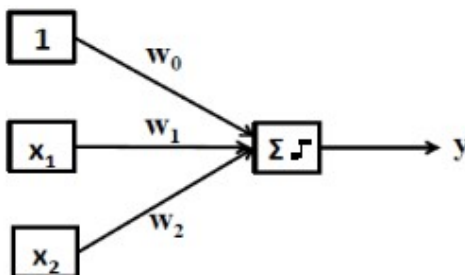


Figure 1

Output of this perceptron is computed using a simple threshold and is given as

$$y = \begin{cases} 1 & \text{if } (w_0 + w_1 \cdot x_1 + w_2 \cdot x_2) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

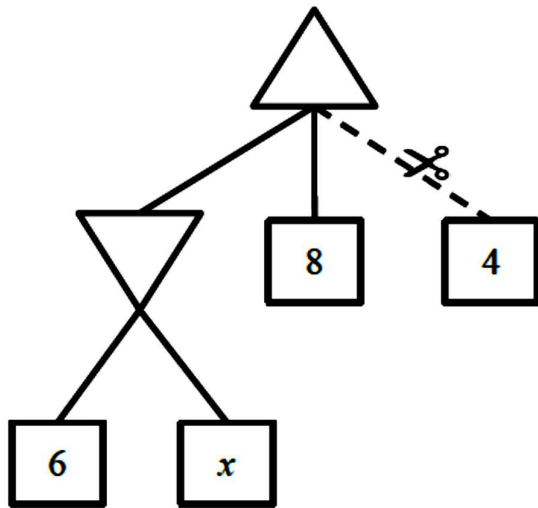
Part a) It is desirable to learn weights  $[w_0 \ w_1 \ w_2]$  of a perceptron that is similar to the perceptron in Figure 1. This perceptron will have two real valued inputs  $x_1$  and  $x_2$  and a bias term set permanently to 1. The objective is to learn weights such that perceptron gives a 1 as output if  $x_2 \geq x_1$  and -1 otherwise. In this part you are required to create training data that can be used to learn weights of the perceptron. The training data must consist of two positive examples (i.e. examples for which output must be 1) and two negative examples (i.e. examples for which output must be -1)

Part b) Use perceptron learning rule to learn weights  $[w_0 \ w_1 \ w_2]$  of the perceptron using your training data of part a. You must start with initial weights  $[0 \ 0 \ 0]$  and only do two iteration/epochs for learning the weights

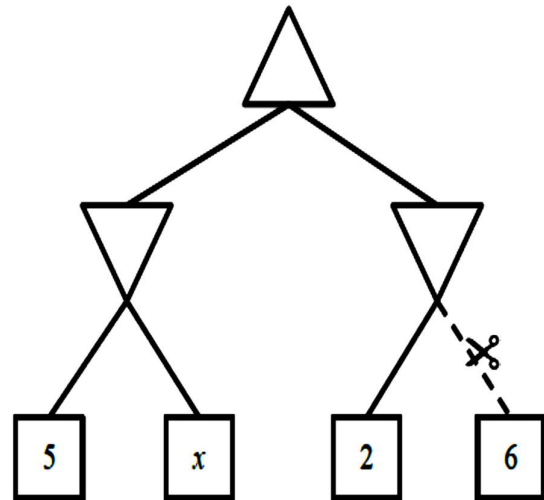
**Question [Game Playing]**

**[1 + 1 + 1 + 1 Points]**

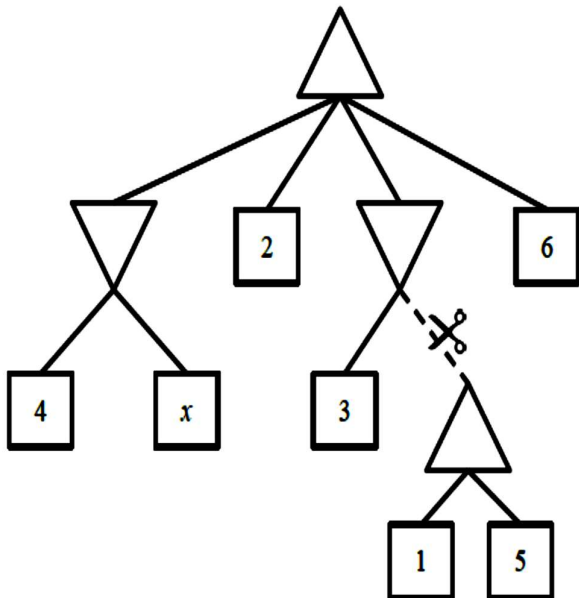
For each of the game-trees shown below, state for which values of  $x$  the dashed branch with the scissors will be pruned. If the pruning will not happen for any value of  $x$  write \none". If pruning will happen for all values of  $x$  write \all".



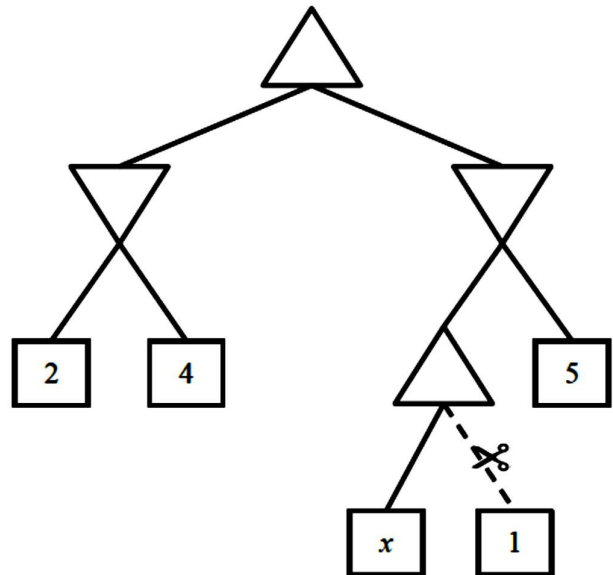
$x$  \_\_\_\_\_



$x$  \_\_\_\_\_



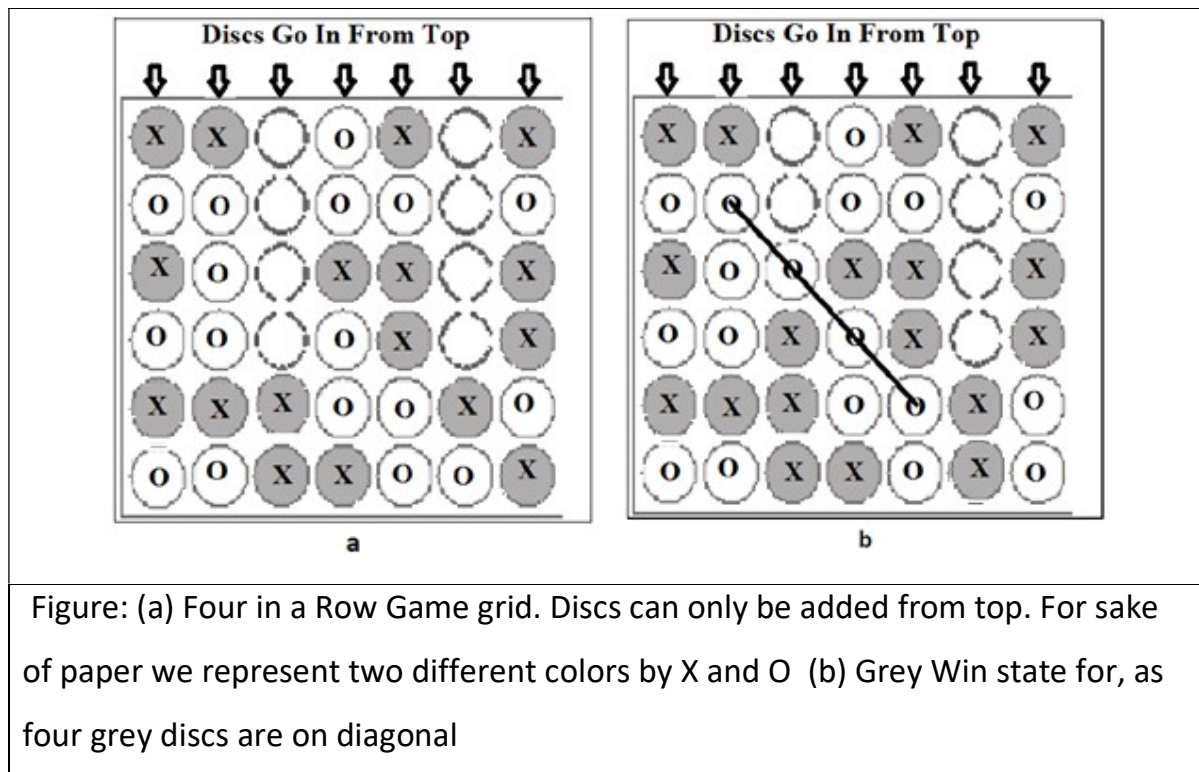
$x$  \_\_\_\_\_



$x$  \_\_\_\_\_

**Question: [Game Playing Again: Four In A Row ]**

Four in a Row is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. (As shown in figure 1a) The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs before your opponent.



The rules for Four in a Row are simple.

- The field (board) has seven columns and six rows.
- Two players play by alternately dropping a chip down one of the columns (from top).
- The chip drops to the lowest unoccupied spot in that column.
- The first player to get four of his own chips in a row, either vertical, horizontal, or diagonal, wins.
- The game ends in a draw if it fills before someone wins.

An AI student has decided to build an automatic player of FOUR IN A ROW using MINIMAX algorithm. Initially he decide to calculate a move at any given point in the game by building a complete game tree.

a) How many nodes will the game tree have when making the first move?  
(Give an approximate Answer) Note that at each level a player has about five possible moves on average [2 Points]

The student figured out that the number of nodes in the game tree is large enough to prohibit building a complete game tree therefore he decided to choose a move by looking only D level deep in the tree. For this purpose he comes up with the following evaluation/expert function E.

```
int[][] evaluationTable = {{3, 4, 5, 7, 5, 4, 3},
                           {4, 6, 8, 10, 8, 6, 4},
                           {5, 8, 11, 13, 11, 8, 5},
                           {5, 8, 11, 13, 11, 8, 5},
                           {4, 6, 8, 10, 8, 6, 4},
                           {3, 4, 5, 7, 5, 4, 3}};
//This evaluation table is used as follows

int evaluateContent() {
    int utility = 128;
    int sum = 0;
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < columns; j++)
            if (board[i][j] == 'O')
                sum -= evaluationTable[i][j];
            else if (board[i][j] == 'X')
                sum += evaluationTable[i][j];
    return utility + sum;
}
```

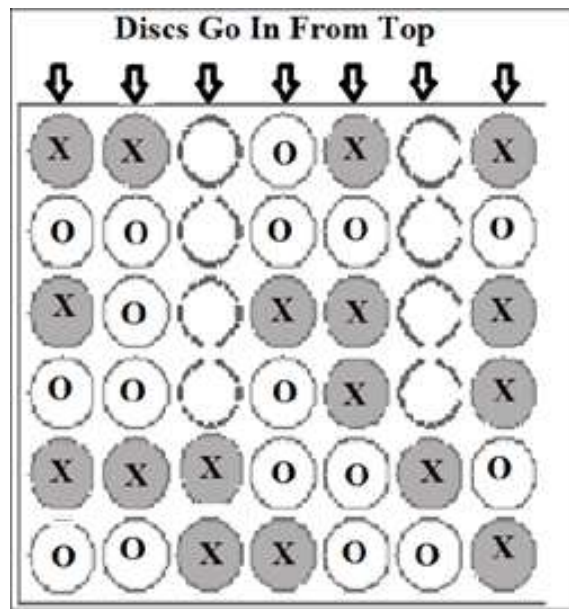
The main idea behind this evaluation function is that the numbers in the table indicate the number of four connected positions which include that space. This gives a measurement of how useful each square is for winning the game and hence it helps decide the strategy. The student implemented the MINIMAX (with alpha-beta pruning) algorithm using his evaluation function. For the state of game given in figure 2.

Part b) For the game state shown below where it is turn of X to make a move show which move will be selected by the MINIMAX if D = 3 i.e. depth of at most 3 is used to select a



move. Show complete game tree used by the player and also show all pruned nodes.

[3 Points]



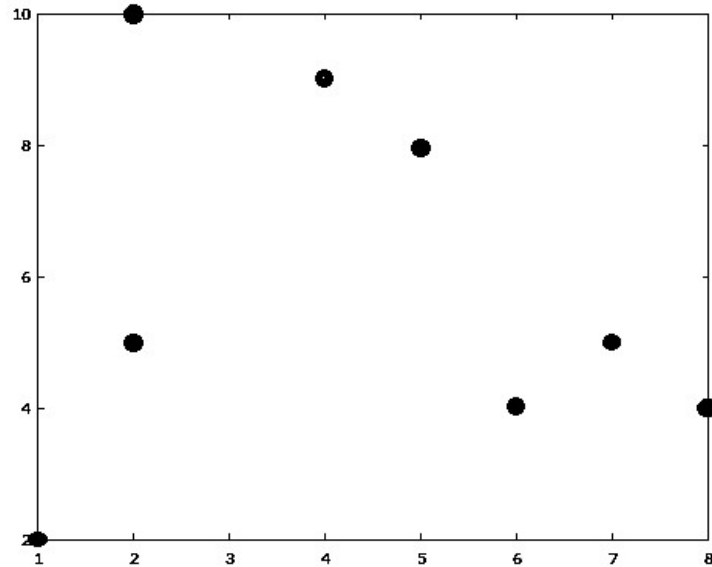
Part c) Repeat part b assuming that it is turn of O and maximum depth is 2

**Question [Clustering]****[2 + 2 + 2 + 2 Points]**

K-Means Clustering For the following points use one iteration of K-means clustering to partition the given points in 3 clusters.

Points: A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9)

For your convenience the a plot of the points is shown below



Assume that the following points are the Initial centers: C1= (2, 10), C2= (5, 8), C3= (1, 2)

i) You have to fill the following tables with the correct values.

	Distance From C1	Distance From C2	Distance From C3
A1(2 10)			
A2(2 5)			
A3 (8 4)			
A4(5 8)			
A5 (7 5)			
A6 (6 4)			
A7 (1 2)			
A8 (4 9)			

ii) Using the distances computed in part i , determine the clusters

Points in first Cluster	Points in Second Cluster	Points in Third Cluster

iii) Use the new clusters to find means of the new clusters

- iv) Finally plot the centers on the figure below and identify the final clusters after the first iteration. [You might not compute all the distances again, just estimate the cluster from the visual distances from the figure.]

[Question 2][NN] [5 + 5 + 5 Points]

CEO of a large IT company assigns either a GOOD (1) or a SATISFACTORY (0) grade to each employee based on their performance throughout the year. CEO want us to automate his grade assigning job by building a fully connected feed-forward network that can assign grades similar to the grades assigned by the CEO.

The CEO uses the following information to assign grades to the employees.

- i) Number of projects/teams the employee worked on throughout the year.
- ii) Average success ratio of the project (a number between -1 and 1).
- iii) Average completion time of each project (number between -1 and 1).
- iv) Skill level of the employee (a number between -1 and 1)
- v) Employees overall experience (a number between 0 and 1)

A teacher at FAST suggested to use the following neural network for mimicking the grade assignment process. The network has one hidden layer of 2 neurons and a single output neuron with linear activation function used in all hidden neurons and sigmoid activation used at output neuron.

**Part a)** How would you prepare training data for training this network? (Remember that the network is being designed to mimic the CEO)

**Part b)** Assuming that the neurons DO NOT have bias term, find the grade assigned to two employee given that

- the employee data for two employees for the year is  
[5, -0.4, 0.8, -0.1, 0.3]  
[1, 0.4, 0.1, 1, -0.5]
- The trained network weights are

Hidden Neuron Weights	Output Neuron Weights
-0.2 0.5 0.5 0.5 0.5 0 1 1 1 1	2 0.1 0.1

**Part c)** Now assume that we have a new training example  $[4, -0.5, 0.3, -0.2, 0.5]$  with a label good available and want to update the existing weights of the neural networks. Using this example Show all intermediate outputs and working to compute the updated weights of the NN

**Part d)** Impressed by the performance of the NN the CEO is now interested in creating a new network that can assign three different grades i) GOOD ii) AVE iii) BELOW AVE. How would you modify the architecture of the neural network proposed above to handle the new demands? Also draw the architecture of resulting neural network.