

# CS 6601 Final – Spring 2021

*Please read the following instructions thoroughly.*

Fill out this PDF form and submit it on [Gradescope](#). Remember to also submit on Canvas. **You will be penalized with 5 points on this exam if you don't submit on both the platforms.**

You have unlimited resubmissions until the deadline. You can: **(a)** type directly into the form – we highly recommend using Adobe Reader DC (or Master PDF on Linux). Other programs may not save your answers, so **please keep a backup**; or **(b)** print, hand-write & scan. You can combine the methods as well.

**Submit only a single PDF** – no phone pictures, please! (You may use an app like CamScanner or Office Lens if you do not have scanner access.) Do not add pages unless absolutely necessary; if you do, please add them at the end of the exam **only**, and clearly label **both** the extra page and the original question page. Submit **ALL** pages of the exam, not only the completed ones.

**Do not forget to fill the checklist at the end before turning in the exam.** The exam may not be graded if it is left blank.

The exam is open-book, open-note, open video lectures, with no time limit aside from the open period. No internet use is allowed, except for e-text versions of the textbook, this semester's CS6601 course materials, Piazza, and any links provided in the PDF itself. No resources outside this semester's 6601 class should be used. Do not discuss the exam on Piazza, Slack, or any other form of communication. More generally, do not post **publicly** about the exam. If there is a question for the teaching staff, **please make it private on Piazza and tag it as Final Exam with the question number in the subject line** (for example, a question on Search would be "Final Exam #2"). Please make **different posts for different questions**.

**Please round all your final answers to 6 decimal places, don't round intermediate results.**

You can use `round(your_number, 6)` function in Python for help.

**You will not receive full credit if your answers are not given to the specified precision.**

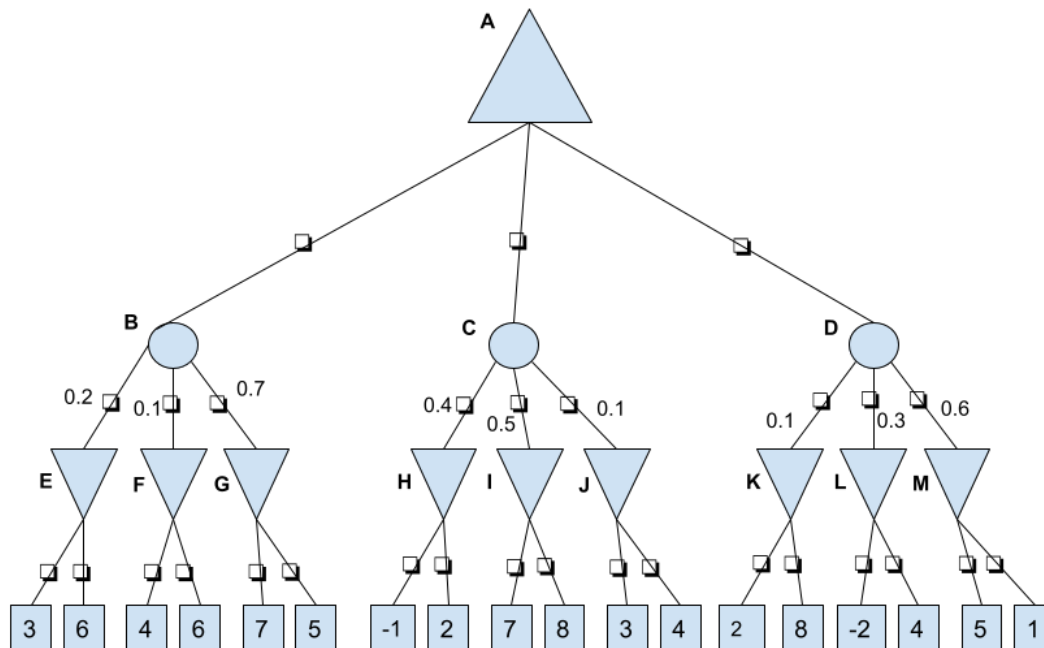
**Point breakdown** (Each question has sub-parts with varying points):

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
Pts	6	6	11	6	8	13	15	15	80

# 1. Game Playing

(6 points)

Solve the game tree below using Expectimax with alpha-beta pruning. Node names are given to the left of each node and probability values are given on the top of individual nodes. The evaluation function is bounded by  $[-8, 8]$ . Evaluate the tree from left to right.



Answer the below questions, using appropriate inequality signs (eg.  $x \leq y \leq z$ ), if needed. If a node is never explored, write its value as 'NE' (without any quotations)

1.1 What is the value of A? (0.5 points)

---

1.2 What is the value of G? (0.5 points)

---

1.3 What is the value of J? (0.5 points)

---

**1.4** What is the value of C? **(1 points)**

---

**1.5** What is the value of L? **(1 points)**

---

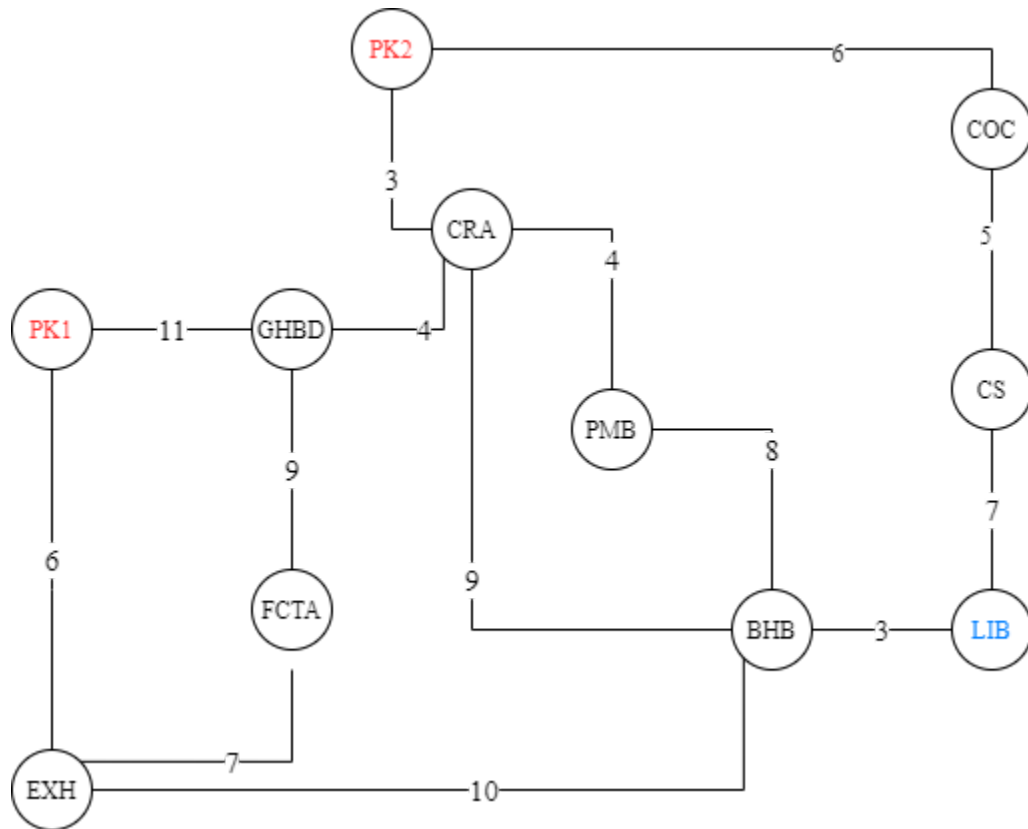
**1.6** What is the value of D? **(1 points)**

---

**1.7** Total number of pruned nodes under the subtree of node B (including B) is \_\_\_\_\_ and under the subtree of node C (including C) is \_\_\_\_\_ and under the subtree of node D (including D) is \_\_\_\_\_. (Make sure to count leaf nodes as well as internal nodes, if any) **(1.5 points)**

## 2. Search

(6 points)



The graph above is a simple GT campus map using abbreviated expressions to denote campus building. The number on each edge is the cost of that edge in terms of distance. For example, the cost from PK1 to GHBD is 11. We have two highlighted nodes **PK1** and **PK2** which stand for Parking-Area 1 and Parking-Area 2. Suppose that you are a student living outside campus, and you plan to go to the library (node LIB highlighted in **Blue**) today by vehicle.

**2.1** Which Parking area would you park in so that you can take the lowest cost path to the library in terms of distance? **(1 Point)**

- ☐ PK1
- ☐ PK2

- 2.2 What are the optimal paths to the library from both PK1 and PK2? Please separate the nodes by comma. For example, the path between PK1 and FCTA is PK1, EXH, FCTA. (2 Points)

From PK1: \_\_\_\_\_

From PK2: \_\_\_\_\_

- 2.3 If we use uniform cost search to find a path from PK1 to the library, please list the third node to be explored during UCS. Then, provide the shortest path and the cost to that third node to be explored. Please follow the convention for path as specified in 2.2. Note that PK1 will be the first node to be explored. (2 Points)

Third Node: \_\_\_\_\_

Cost: \_\_\_\_\_

Path: \_\_\_\_\_

- 2.4 Recall that for A\* search algorithm,  $f(x) = g(x) + h(x)$  where  $h(x)$  is the heuristic function.  $g(x)$  is the cost of the edge in our graph. Let  $d(x)$  be the euclidean distance heuristic function (assume for this question that  $d(x)$  is always admissible). Which of the following heuristics might be inadmissible? Select all that apply. (1 Point)

- ☐  $h(x) = \max(4, \log(d(x)))$
- ☐  $h(x) = 1.5 * d(x)$
- ☐  $h(x) = d(x) * d(x)$
- ☐  $h(x) = d(x) + 50$
- ☐  $h(x) = \max(0, d(x) - 50)$

### 3. Optimization Algorithms

(11 points)

#### Background

A standard optimization problem is defined by two elements: an **objective function** and **constraints**. The objective function is typically a function for which you want to find extrema. The constraints specify the range of the domain of your search, as most searches don't extend into an infinite domain. If you have an optimization problem and can formulate a clear objective function and constraints, you're in a position to apply a wide range of optimization methods to find your answer such as genetic algorithms, simulated annealing, stochastic beam search, or gradient descent. One example of a standard formulation of an optimization problem is:

$$\begin{aligned} \min \quad & y - ax \\ \text{subject to} \quad & x + 2y \leq 4 \\ & -y + 2x \leq 3 \\ & x \geq 0, y \geq 0 \end{aligned}$$

#### Problem

Arya is a graduate student at Georgia Tech who is working closely in measuring her stress levels by quantifying its linear relationship using passively sensed behaviours. She is using her Fitbit Flex 2 and a downloaded mobile application to track her steps, sleep and other passively sensed information. She aggregates the data she has collected on a daily basis over 5 days and designs some features  $(x_1, x_2, x_3)$  to predict her stress levels. Arya relies on the optimization algorithms that she learnt in her AI class and decides to formulate this problem as a gradient descent problem.

Her objective is to come up with the best set of parameters for her designed features that allow her to predict her stress levels. In order to do so she utilizes the Gradient Descent optimization technique to minimize the error between her predicted stress levels and her actual stress levels. She makes use of the squared loss function that is summed over all the data points in the given dataset:

$$\text{Error function} = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$y_i$  = The measured stress level

$\hat{y}_i$  = The predicted stress level

$n$  = number of datapoints in the dataset.

$$\text{Predicted Stress Level } (\hat{y}_i) = ax_1 + bx_2 + cx_3 + d$$

$a, b, c, d$  = parameters

$x_1, x_2, x_3$  = features

Datapoint	$x_1$	$x_2$	$x_3$	$y$
1	12	6	3	4
2	5.5	8.3	5	2
3	4.7	4.5	2	5
4	15.5	7.3	9	3
5	3	7	7	2

**Table 4.1**

She summarizes her data in Table 4.1 and she hopes to solve this problem using the fundamentals of gradient descent. However, the parameters ( $a, b, c, d$ ) are unknown and to calculate these parameters Arya must traverse the space using the following gradient descent algorithm:

1. Start at an initial setting of parameters = ( $a, b, c, d$ ) and a step size  $s$ .
2. Calculate the predicted stress for each datapoint in the dataset using the current parameter setting.
3. Calculate the gradient of the error function for each datapoint in the dataset with respect to each parameter  $\delta(a), \delta(b), \delta(c), \delta(d)$ .
4. Update the parameters by moving to the position  
 $(a - s \times \delta(a), b - s \times \delta(b), c - s \times \delta(c), d - s \times \delta(d))$
5. Repeat steps 2,3,4 until the parameters do not significantly reduce the error

## Gradient Evaluation

The gradient of the error function with respect a and b are given below where  $n$  is the total number of datapoints in the dataset:

$$\delta(a) = - \sum_{i=1}^n x_{1,i} (y_i - \hat{y}_i) \quad \delta(b) = - \sum_{i=1}^n x_{2,i} (y_i - \hat{y}_i)$$

**3.1.** What is the gradient of the error function with respect to c and d? **(1 point)**

- ☐  $\delta(c) = \sum_{i=1}^n x_{3,i} (y_i - \hat{y}_i), \delta(d) = \sum_{i=1}^n (y_i - \hat{y}_i)$
- ☐  $\delta(c) = - \sum_{i=1}^n x_{3,i} (y_i - \hat{y}_i), \delta(d) = \sum_{i=1}^n (y_i - \hat{y}_i)$
- ☐  $\delta(c) = - \sum_{i=1}^n x_{3,i} (y_i - \hat{y}_i), \delta(d) = - \sum_{i=1}^n (y_i - \hat{y}_i)$
- ☐ None of the above

**3.2.** Use the algorithm described above to perform gradient descent for 2 iterations. Please remember to round your answers as per the rounding rules of the exam. **(4 points)**

**step size (s) = 0.1**

**initial parameters (a,b,c,d) = (0.1, 0.2, 0.5, 1)**

Iteration	$\delta(a)$	$\delta(b)$	$\delta(c)$	$\delta(d)$
1				
2				

**3.3.** Fill in the missing parameter values that you obtain after completing the two iterations of the gradient descent algorithm? You should use the values provided to validate the correctness of your answer. Please remember to round your answers as per the rounding rules of the exam. **(3 points)**

a = 1017.720429

b = \_\_\_\_\_



c = \_\_\_\_\_

d = 106.378954

So far, you carried out a variant of the gradient descent algorithm called **batch gradient descent**. In this method the parameters are updated by taking the gradient over the entire dataset. This might prove inefficient in cases where the dataset is large. To alleviate this issue, there exists a second variant of the algorithm called **stochastic gradient descent** in which the parameters are updated after evaluating the gradient over a single datapoint as opposed to the entire dataset.

- 3.4.** Apply stochastic gradient descent approach to fill in the values of the parameters at each iteration. Note that at each iteration the gradient is calculated over a single datapoint to update the parameters i.e at iteration 1 calculate the gradient over datapoint 1, update the parameters, and continue similarly. Please remember to round your answers as per the rounding rules of the exam. **(3 points)**

**step size = 0.1**

**initial parameters (a,b,c,d) = (0.1, 0.2, 0.5, 1)**

Iteration	a	b	c	d
1	-0.98	-0.34		
2			4.306	1.7252
3	-20.335665	-16.398668		

## 4. Constraint Satisfaction Problems

(6 points)



Quantum Wire & Cable Corporation has seen a flood of new product orders as the economic stimulus takes hold. However, the software system that was responsible for assigning jobs to cable extrusion lines needs updating.

As Thad's student of AI, you know that industrial scheduling is often a constraint satisfaction problem. You decide to manually assign production assets for incoming jobs while waiting for a response to your software service ticket.

Right now, there are five different production jobs to schedule across four different extruder machines: each job is to be scheduled to one extruder. Some but not all jobs may be assigned to the same extruder. Here are the details:

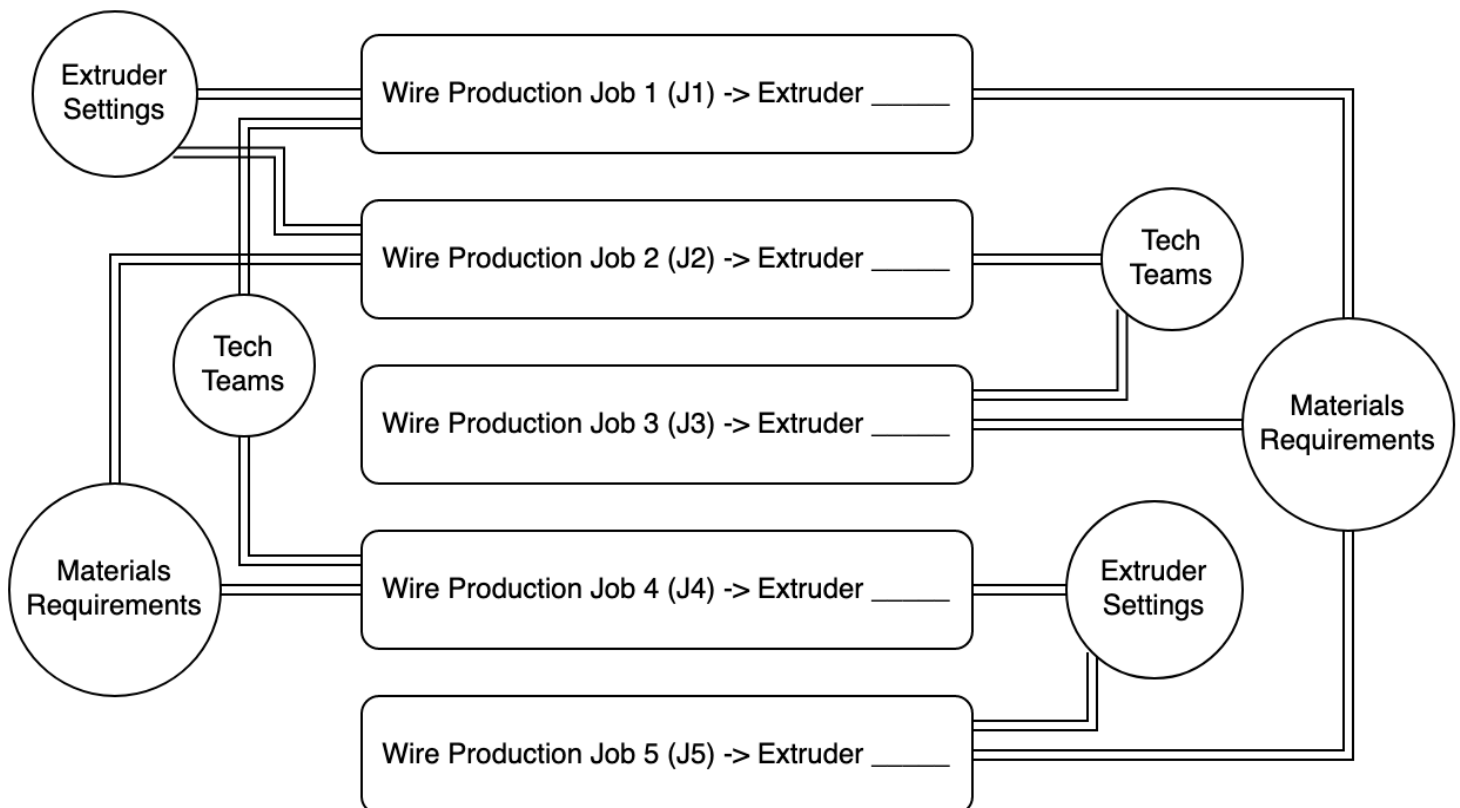
1. The production jobs are labeled J1, J2, J3, J4, J5
2. Each of those requests can be allotted to one of the extruders - E1, E2, E3, E4
3. Jobs J1, J3, and J5 have different materials requirements and thus cannot be scheduled to the same extruder. The same applies for J2 and J4.
4. Some extruder settings can't be changed mid-job, thus J1 and J2 cannot be assigned to the same extruder. The same applies for jobs J4 and J5.
5. Technical teams on the factory floor need to devote full attention to specific extruders; therefore, J2 and J3 cannot be assigned to the same extruder, nor can J1 and J4 be assigned to the same extruder.

You have decided to implement a constraint graph over the production job variables J1, J2, J3, J4, J5 in order to aid the factory in extruder assignment. Then you decided to solve the problem by implementing backtracking. Here are the details of that implementation:

1. You choose unassigned variables using the MRV heuristic. You break ties by choosing the variable with the lower index, i.e., J1 chosen over J3 if both have the same remaining values.
2. When ordering domain values, you choose the value that has the lowest index. So, if E1 and E3 are valid domain values, you pick E1.
3. You perform inference using Forward Checking.

You follow this paradigm, filling out the domains for all the variables after each step of the backtracking search. You stop when all the variables have an assigned value, or if an assignment is not possible.

- 4.1** Using this paradigm, complete the diagram below by filling in the blank for the extruder each job is assigned to. **(4 points)**



- 4.2** In addition to the change in product requirements, you stumbled on a new extrusion materials vendor, removing the materials requirements constraints. Is it possible to schedule the jobs now? If so, give a valid assignment, or if not, state there is no valid assignment. **(2 Points)**

Job 1 (J1) - Extruder \_\_\_\_\_

Job 2 (J2) - Extruder \_\_\_\_\_

Job 3 (J3) - Extruder \_\_\_\_\_

Job 4 (J4) - Extruder \_\_\_\_\_

Job 5 (J5) - Extruder \_\_\_\_\_

## 5. Bayes Nets

(8 points)

### Gene Regulatory Networks

Advances in computational biology over the last 20 years have led to a deeper understanding of how genes are expressed. There are several factors that influence whether a gene will be expressed or not - proteins, neighbouring or far off sections of the same gene sequence, relative shapes and positions of molecules. A gene regulatory network (GRN) defines the relationships between these various factors and their interactions. With data for a specific disease, GRNs can help identify disease regulating proteins and genes, which can serve as potential targets for drug development/treatment.

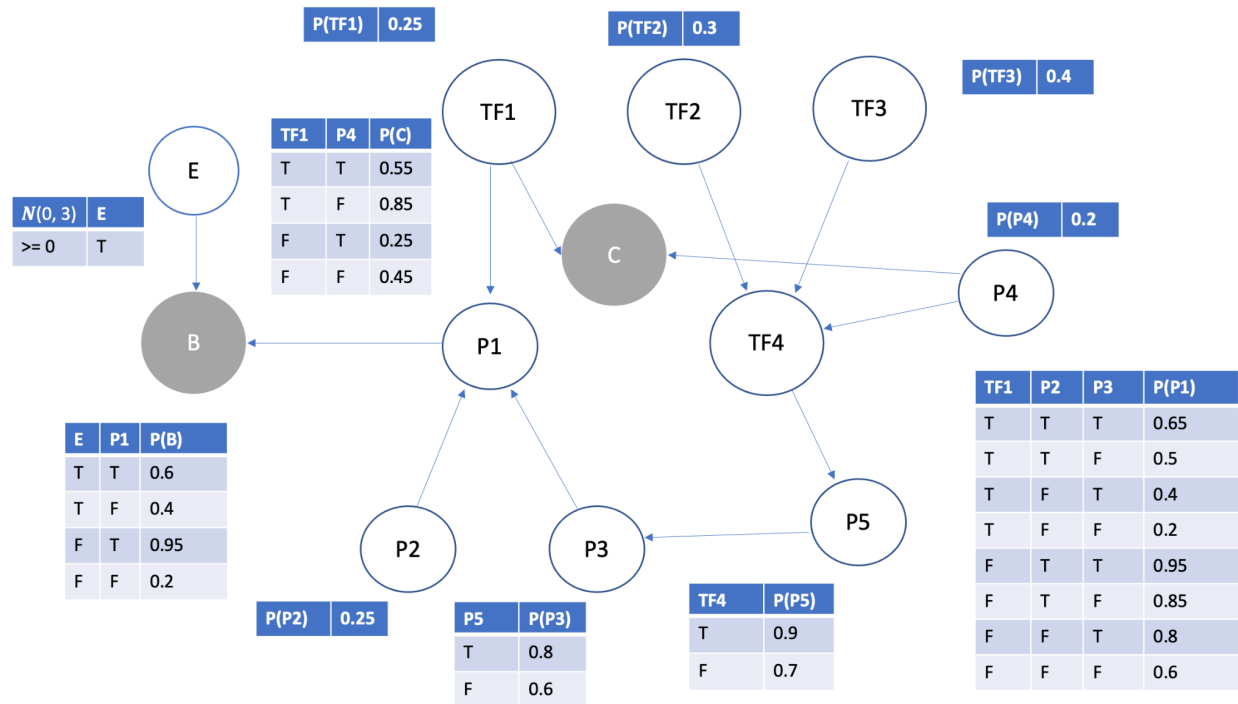
An essential tool in this field of research is Bayesian Networks. They are very flexible for large scale data, and since they learn the distribution for variables instead of value prediction for a specific task, they can absorb information from multiple datasets based on similar diseases/conditions.

Basic factors affecting gene expression:

- *Enhancer Gene*: A sequence of DNA that enhances gene expression (can affect how aggressive the expression will be).
- *Promoter Gene*: A sequence of DNA that initiates the process of gene expression (usually has to be close to the gene to be expressed).
- *Transcription Factor (TF)*: A category of proteins that coordinate with each other to regulate gene expression, it can have a positive or negative influence on gene expression. Multiple TFs can influence a target gene and a single TF can influence multiple target genes.

To get a better picture of how this works, we will look at a case study of a regulatory network to predict the occurrence of a rare blood disorder. Instead of dealing with enhancer/promoter relationships between genes, we will be dealing with the proteins related to them instead.

In the Bayes Net below, B is a seemingly harmless protein found in Red Blood Cells, but whose concentration can be measured. C is a gene that if expressed leads to a rare blood disorder, but cannot be measured directly. To keep this simple, we are only worried whether a protein/transcription factor is present or not and only if a gene is expressed or not i.e., they are all binary variables. There is a component of random noise associated with measuring the presence of B though, in the form of random gaussian noise  $N(0, 3)$  (mean 0, std deviation 3)



P - protein  
 TF - transcription factor  
 E - random gaussian noise  
 B - protein found in RBCs  
 C - blood disorder causing gene

5.1. Which of the following statements are true? Independence is represented as in the format in this example:  $A \perp B \mid C$  means that A is independent of B given C. Mark all that apply. (3 points)

- ☐  $B \perp C \mid TF1$
- ☐  $P1 \perp P4 \mid P5$
- ☐  $TF1 \perp P3 \mid P2, B$
- ☐ The intersection of the markov blankets of P4 and P3 is a null set.
- ☐ The random noise for the observation of B (E) doesn't affect P(C) given P3 and P4 i.e., is  $C \perp E \mid P3, P4$
- ☐ P1 is in the markov blanket of P5

- 5.2. Given a person has the protein (B=true), what is the probability of that person expressing the blood disorder C given that P2 and P3 are not present i.e.,  $P(C | B, \neg P2, \neg P3)$ ? You must show your work below. You cannot use a Bayesian Inference API to solve this (whether it was written by you or someone else). Please remember to round your answers as per the rounding rules of the exam. (5 points)

Provide some **brief** reasoning to your answers in 5.2.

Because P3 is given, TF2, TF3, TF4 and P5 is useless.

Using Variable Elimination:

$$\begin{aligned}
 & P(C | B, \neg P2, \neg P3) \\
 &= \alpha \sum_E \sum_{P1} \sum_{TF1} \sum_{P4} P(E) P(B | E, P1) P(P1 | \neg P2, \neg P3, TF1) P(C | P4, TF1) P(TF1) P(\neg P2) P(\neg P3) P(P4) \\
 &= \alpha \sum_E P(E) \sum_{P1} P(B | E, P1) P(\neg P2) P(\neg P3) \sum_{TF1} P(P1 | \neg P2, \neg P3, TF1) P(TF1) \sum_{P4} P(C | P4, TF1) P(P4) \\
 &= \alpha \sum_E f(E) \sum_{P1} f(E, P1) \sum_{TF1} f(P1, TF1) f(TF1) \sum_{P4} f(C, P4, TF1) f(P4) \\
 &= \alpha \sum_E f(E) \sum_{P1} f(E, P1) \sum_{TF1} f(P1, TF1) f(TF1) f(C, TF1) \\
 &= \alpha \sum_E f(E) \sum_{P1} f(E, P1) f(P1, C) \\
 &= \alpha \sum_E f(E) f(E, C) \\
 &= \alpha f(C)
 \end{aligned}$$

	\	+C	-C
$f(C, TF1) =$	+TF1	0.79	0.21
	-TF1	0.41	0.59

	\	+C	-C
$f(P1, C) =$	+P1	0.224	0.276
	-P1	0.281	0.219

	\	+C	-C
$f(E, C) =$	+E	0.2468	0.2532
	-E	0.269	0.306

	\	+C	-C
$f(C) =$	+C	0.2579	0.2796
	-C		

Normalize  $f(C)$ ,

$$P(C | B, \neg P2, \neg P3) = 0.479814$$

## 6. Machine Learning

(13 points)

Deep learning has seen explosive growth in the last decade. Since the introduction of AlexNet in 2012, we have advanced from being able to tell a picture of a dog and a cat apart to having computer vision guide self-driving cars. The backbone of this revolution is the Convolutional Neural Network, or CNN, a particular type of neural network that was designed specifically for computer vision and was partially inspired by the human visual cortex. A CNN is able to learn spatial information about its input images through the use of filters. These filters are learnable parameters in the network that can be trained to identify a variety of features, from edges or shapes in higher levels of the network to more complex features like eyes or car tires in deeper layers of the network.

A CNN is able to extract information about input images by using the convolution operation. Convolutions are used in a variety of fields and can be described in slightly different ways depending on the application. For the purposes of CNNs we will define the convolution operation as follows. Given an  $N \times M$  image and a  $k \times k$  kernel, the convolutional operation will “slide” the kernel across the image and multiply the kernel values element wise with the corresponding image values that fall within the kernel window. It will then sum all of the corresponding values and use this sum as the resulting output value for that window. It can be described formally as:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n)K(m, n)$$

See the picture below for a visual explanation of the operation, using the following kernel matrix:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

In the example above, the image values are shown in green. The kernel values are given by the red subscripts. The convolution window is highlighted in yellow. The image values are first multiplied by the kernel values, after which they are summed, resulting in 4. A few more steps of the convolution are shown below.

1	1 <sub>x1</sub>	1 <sub>x0</sub>	0 <sub>x1</sub>	0
0	1 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	0
0	0 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved  
Feature

1	1	1	0	0
0	1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0
0	0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1
0	0 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0
0	1	1	0	0

Image

4	3	4
2		

Convolved  
Feature

1	1	1	0	0
0	1	1	1	0
0	0	1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>
0	0	1 <sub>x0</sub>	1 <sub>x1</sub>	0 <sub>x0</sub>
0	1	1 <sub>x1</sub>	0 <sub>x0</sub>	0 <sub>x1</sub>

Image

4	3	4
2	4	3
2	3	4

Convolved  
Feature

In this question, we are going to develop a deeper intuition regarding how the convolution operation works.

For all parts of this question, please record any answers which are matrices in **row major order** using commas to separate your values. See the image below for an example of row major order:



### Row-major order

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Row major order for this matrix would be  $a_{11}, a_{12}, a_{13}, a_{21}, \dots, a_{33}$

### Convolution with Padding

As you've seen in the previous example, the convolution operator can reduce the size of the input. A simple but effective way to deal with this is by using zero padding. Zero padding simply appends zeros at the ends of the input image. For example, if the matrix on the left represents an image, applying a zero padding of 1 will produce the matrix on the right.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Consider the following kernel:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 3 & 2 & 3 \end{pmatrix}$$

If we convolve this kernel with the padded input above, we get the following result:

$$\begin{pmatrix} 7 & 13 & 7 \\ 9 & 10 & 9 \\ 4 & 5 & 4 \end{pmatrix}$$

As you can see, the output has retained the same size as the input due to the zero padding.

## Question A - Multi-Filter Convolution

(7 points)

One of the strengths of a convolutional neural network is that each convolutional layer in the network can apply multiple filters to a single input, thus learning multiple features of the input at each layer. Consider the following input and kernels:

Input:

$$\begin{pmatrix} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 50 \\ 10 & 10 & 10 & 50 & 50 \\ 10 & 10 & 50 & 50 & 50 \\ 10 & 50 & 50 & 50 & 50 \end{pmatrix}$$

$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}$$

Kernel 1

$$\begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}$$

Kernel 2

$$\begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix}$$

Kernel 3

Convolve the three kernels across the input using a **zero padding of 1**. Report all your answers in the row-major order.

**6.A.1** Result after convolving with Kernel 1. (2 points)

---

**6.A.2** Result after convolving with Kernel 2. (2 points)

---

**6.A.3** Result after convolving with Kernel 3. (2 points)

---

**6.A.4** Based on your results, what kind of features do the above filters find? (1 points)

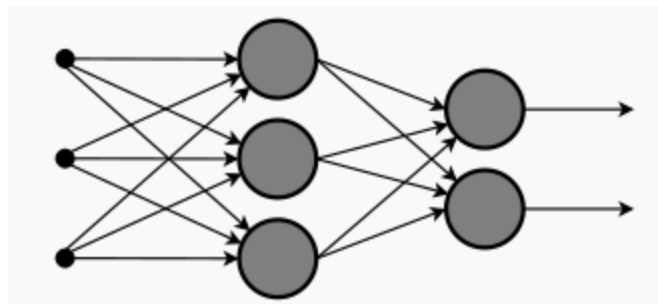
- ☐ The first filter is a diagonal line detector, the second filter is a horizontal line detector and the third filter is an edge detector
- ☐ The first filter is a horizontal line detector, the second filter is a vertical line detector and the third filter is a positive slope line detector
- ☐ The first filter is an edge detector, the second filter is a vertical edge detector and the third filter is a horizontal edge detector
- ☐ The first filter is a vertical line detector, the second filter is a horizontal line detector and the third filter is a negative slope line detector

## Question B - Convolutions, Backprop and Gradient Descent

(6 points)

Until now, we have used predefined filters to perform our convolutions. However, in a real CNN, these kernels are the learned features of the network. Before we discuss how we can learn filters in a CNN, let's discuss how neural networks learn generally.

A traditional neural network might look like the one below. The black dots are inputs and the large gray dots are nodes. Each arrow in between nodes are the weights of the network. The weights determine how much each previous input affects the calculation for the current node. Each node performs some calculation on its inputs and produces some output. The output of the final layer is usually what we are interested in most, since this output is the desired value we are trying to predict, whether this be stock prices or what type of image the network just looked at.



How does the network know if it predicted a good value? During training, we provide the network with labeled examples. After passing each training example through the network, we compare its predicted result with the label of the example. In order to quantify the “correctness” of the prediction, we use a loss function, which quantifies how far off the network’s prediction was from the actual label. If the network made a good prediction, the loss is small. If it was far off the mark, the loss is large. After finding this loss, we need to propagate this loss back through the network and update the parameters of the network, in this case the weights, so that

the next time the network sees this example, it makes a better prediction. This process is known as backpropagation.

How does all of this apply to CNNs? The filters in the CNN are, in fact, the weights of the network! When performing backpropagation through a CNN, you are actually tuning each filter in the network to be able to pick out better features in your input images so that the network can classify them more effectively. Let's walk through an example.

### Forward Pass of the network

Consider the following input and kernel:

Input:

$$\begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

Kernel:

$$\begin{pmatrix} 12 & 39 \\ 9 & 12 \end{pmatrix}$$

Our goal is to be able to detect the diagonal of the matrix. We will know we've done this if the output of our convolution has large values corresponding to the diagonal and small or zero value everywhere else.

**6.B.1** Convolve this kernel across this input. Do not use zero padding for this convolution. Remember to report your answer in the row-major order. **(1 points)**

---

### Backpropagation and Gradient Descent

The output is pretty noisy. How can we fix this?

As mentioned previously, backpropagation is the process by which the neural network updates its weights, in this case the filters, to better classify its inputs. Intuitively, this process can be thought of as learning from error. Once the loss, or error, is calculated, we want to discover how each parameter of the network affected this loss value.

In order to do this, we first calculate the derivative of the loss with respect to the output of the network and then propagate this back by using the chain rule to calculate the derivative of the loss with respect to every parameter in the network. This can be a messy process, but the good

news is that in a CNN, we can solve this problem using convolutions! Given the derivative of the loss with respect to the output, which is a matrix in this question, we can calculate the derivative of the filter with respect to the loss by convolving the loss derivative with the input. The resulting operation will be a matrix of the derivative of each weight in the filter with respect to the loss.

Once we have these values, we can use a process known as gradient descent to update the parameters of the network. Simply put, gradient descent tries to minimize the loss function by updating the parameters of the network. In order to do this, we use the derivative of the loss with respect to the parameters we calculated in backpropagation and use a new variable,  $\eta$ , which is the learning rate. The learning rate controls how much to change each parameter in each backwards pass of the network. The equation for gradient descent is as follows:

$$w_{n+1} = w_n - \eta * \frac{\partial L}{\partial w_i}$$

For the following question, assume that we have the following matrix of values, which correspond to the derivative of the loss with respect to the output:

$$\begin{pmatrix} 2 & -1 \\ 4 & -3 \end{pmatrix}$$

**6.B.2** Calculate the matrix of the derivative of the filter with respect to the loss. Report your answer in row major order. Do not use padding for this convolution. **(1 points)**

---

**6.B.3** Using your answer from the previous question, perform gradient descent on the filter and report the new filter values in row major order. **(2 points)**

---

**6.B.4** Finally, convolve the new filter with the input and report your answer here in row major order. Do not use padding for this convolution. **(2 points)**

---

As you can see, the result is much less noisy now! Typically, a CNN would continue training for hundreds of iterations learning better and better filters until it is able to classify its input near perfectly. Hint: Use this insight to validate that you've gotten the correct values. There may not be partial credit awarded for these questions (6.B.2, 6.B.3 and 6.B.4).

## 7. Logic and Planning

(15 points)

### Question A - Logic

(8 points)

7.A.1 Mark all the correct statements from below. (2.5 points)

- ☐ Inference in first-order logic can be made if it can be reduced to propositional logic.
- ☐ A complete inference algorithm may not be sound.
- ☐ Soundness is to completeness in logic as recall is to precision in machine learning.
- ☐ Logical sentences have to be in Conjunctive Normal Form before we use any inference algorithms.

7.A.2 Alex, Bethany, and Cynthia are three suspects for murder.

Alex says *"I did not do it. The victim has been Cynthia's friend. But Bethany hated them."*

Bethany states *"I did not do it. I saw the other two suspects hanging out in the town's park with the victim on the day of the murder; one of them must have done it."*

Cynthia says *"I did not do it and have never known the victim. Besides, I was out of town all the week."*

Assume that the two innocent suspects are telling the truth, but the guilty one might not be. Based on the given information, which of the following option holds?

(Hint: use propositional resolution) (3.5 points)

- ☐ Alex is the murderer
- ☐ Bethany is the murderer
- ☐ Cynthia is the murderer
- ☐ Bethany and Cynthia are innocent but no conclusion can be made for Alex
- ☐ Alex and Cynthia are innocent but no conclusion can be made for Bethany
- ☐ Alex and Bethany are innocent but no conclusion can be made for Cynthia
- ☐ The murderer cannot be found out based on the provided information

**7.A.3** Every kitty chases some rat.

Every rat who eats cheese is smart.

No kitty catches any smart rat.

Any kitty who chases some rat but does not catch it is frustrated.

Using the above information, which of the following statement(s) is/are correct? **(2 points)**

- ☐ If all rats eat cheese, then not all kitties will be frustrated.
- ☐ If all rats eat cheese, then all kitties will be frustrated.
- ☐ If no rats eat cheese, then all kitties are not frustrated.
- ☐ If no rats eat cheese, then some kitties are frustrated.

## Question B - Planning

**(7 points)**

The following portion of a PDDL(Planning Domain Definition Language) description represents a rover collecting samples while exploring a new planet. The rover has a box to store the collected samples.

Types: <Subtype - Supertype> (a type with no Superclass is the Supertype of all types)

Object

Box - Object

Location - Object

Rover - Object

Sample - Object

SampleSpot - Location

Predicates:

Box(b) - returns True if b is a Box

Location(l) - returns True if l is a Location

Object(o) - returns True if o is an Object

Rover(r) - returns True if r is a Rover

Sample(s) - returns True if s is a sample

SampleSpot(ss) - returns True if ss is a SampleSpot

At(o, loc) - returns True iff o is an Object, loc is a Location, and o is at loc

In(o1, o2) - returns True iff o1 and o2 are Objects, and o1 is in o2

**Init(SampleSpot(ss1) ^ SampleSpot(ss2) ^ SampleSpot(ss3) ^ Box(b) ^ Rover(r) ^  
Sample(s1) ^ Sample(s2) ^ Sample(s3) ^ At(b, ss1) ^ At(r, ss1) ^ At(s1, ss1) ^ At(s2, ss2)  
^ At(s3, ss3))**

**Goal(In(s1, b) ^ In(s2, b) ^ In(s3, b) )**

You need to add the following action to your PDDL, **PutInBox(r, b, s, ss)** which is an act of putting a sample into the rover's box, assuming that sample **s** is found at the sample spot **ss** (**r** = rover, **s** = sample, **b** = box, **ss** = sample spot). Also assume that the rover leaves the location after putting the sample in the box.

PRECOND:  $\text{Rover}(r) \wedge \text{Box}(b) \wedge \text{Sample}(s) \wedge \text{SampleSpot}(ss) \wedge \text{At}(s, ss) \wedge \text{At}(b, ss) \wedge \text{At}(r, ss)$

**7.B.1** Choose all the applicable clauses for the **EFFECT**. (3 points)

- ☐  $\text{In}(s, b)$
- ☐  $\sim \text{In}(s, b)$
- ☐  $\sim \text{In}(s, ss)$
- ☐  $\sim \text{At}(s, ss)$
- ☐  $\text{At}(s, ss)$
- ☐  $\text{At}(s, b)$

You need to add the following action to your PDDL, **Transition(b, r, ss1, ss2)** wherein the rover **r** (with box **b**) moves from one sample spot (**ss1**) to another sample spot (**ss2**)

EFFECT:  $\text{At}(b, ss2) \wedge \text{At}(r, ss2)$

**7.B.2** Choose all the applicable clauses for **PRECOND**. (4 points)

- ☐  $\text{Box}(b)$
- ☐  $\text{Rover}(r)$
- ☐  $\text{Sample}(s1)$
- ☐  $\text{SampleSpot}(ss1)$
- ☐  $\text{Sample}(s2)$
- ☐  $\text{SampleSpot}(ss2)$
- ☐  $\text{At}(b, ss2)$
- ☐  $\text{In}(ss1, b)$
- ☐  $\text{In}(ss2, b)$
- ☐  $\text{At}(b, ss1)$
- ☐  $\sim \text{At}(b, ss1)$
- ☐  $\text{At}(r, ss1)$
- ☐  $\sim \text{At}(r, ss1)$
- ☐  $\text{At}(r, ss2)$





## 8. Planning Under Uncertainty

(15 points)

### Initial Environment

The robot is at a start state A3 and has to navigate to the goal state D1 without crashing into obstacles. There are two obstacles in the environment, a chair and a television, that the robot cannot pass through. Moreover, the robot cannot leave the environment. For example, if the robot is at state D3, and takes an action **LEFT**, it will stay in the same state D3 unless the robot unintentionally takes an action perpendicular to the intended action (this will be explained in the **Description of the Actions** section). The robot can go through all the other states but some states are harder to pass through than others. i.e.) State B3's initial utility is -100. The utility of Goal state D1 is 100.

	A	B	C	D
1				Goal
2	-50			-25
3	Start	-100		
4				

### Description of the Actions

The robot can take four actions: UP, DOWN, LEFT, RIGHT. However, the robot has 80% chance of taking the intended action and 10% chance **each** of taking the **actions perpendicular** to the intended action. i.e) if the intended action is UP, there is 80% chance the robot will move UP 10% chance to move RIGHT, and 10% chance to move LEFT. If an action is blocked by the obstacle or environment boundary the robot can still take the action but it will stay in the same state. In this case there still exists a 10% chance **each** of taking the actions perpendicular to the intended action, and the robot will still receive -10 negative reward. For example, if the intended action is UP in state A3, there is 80% chance the robot will move to A2, 10% chance to move RIGHT and reach state B3, and 10% chance to fail to move LEFT and stay in A3.

### Value Iteration

- A robot taking an action will cost -10 reward.(even if the robot ends up in the same state)
- Assume discount factor of 0.5
- At the initial state, the utility of all states is 0 except for states that have negative utilities and the goal state.

**8.1** What is the utility of the below states after the first iteration? Please remember to round your answers as per the rounding rules of the exam. **(6.25 points)**

A1: \_\_\_\_\_

C1: \_\_\_\_\_

D2: \_\_\_\_\_

**8.2** What is the optimal action at state C2 after the first iteration? **(1.25 points)**

\_\_\_\_\_

**8.3** What is the utility of the below states after the second iteration? Please remember to round your answers as per the rounding rules of the exam. We have provided the utility of A2 and B2 for you to verify your answer. **(6.25 points)**

A1: \_\_\_\_\_

A2: -66.125

B2: -18.5

C1: \_\_\_\_\_

D2: \_\_\_\_\_

**8.4** What is the optimal action at state C2 after the second iteration? **(1.25 points)**

\_\_\_\_\_

## 9. Extra Credit

(Up to 2 points)

**CIOS! We will be awarding extra credit to the entire class based on CIOS completion rates.**

The completion rate for CIOS is defined as follows:

$$(number\ of\ students\ who\ completed\ CIOS) / (number\ of\ students\ enrolled\ in\ the\ class)$$

The completion rate (up to 1.0) will be multiplied by 2 and added to the score of your final exam.

Please consider taking the CIOS! As we have said over the course of the semester, this course is constantly evolving and we are looking at ways of making it better. We take CIOS feedback very seriously and incorporate it into our teaching methods wherever possible.

## Checklist

And now mark the checklist below making sure you have taken care of each of the points mentioned:

- ☐ I have read the pinned Piazza post with the title 'Final Exam Clarifications Thread', and I am familiar with all of the clarifications made by the Teaching staff.
- ☐ All answers with more than 6 digits after the decimal point have been rounded to 6 decimal places.
- ☐ All pages are being uploaded in the correct order that they were presented to me, and none of the pages are missing/removed.
- ☐ Any extra pages (**including blanks**) are only attached at the END of this exam, after page 61 with clear pointers to wherever the actual answer is in the PDF (reference properly).
- ☐ I am submitting only one PDF and nothing else (no docx, doc, etc.).
- ☐ The PDF I am submitting is not blank (unless I want it to be).
- ☐ **I will go over the uploaded pictures on Gradescope and make sure that all the answers are clearly visible. I acknowledge that I am aware that dull / illegible / uneven scans will not be graded.**
- ☐ I have submitted a copy of the PDF to Canvas.