Question 1: Logic Programming [10 marks in total]

Consider the following statements:

- We will go swimming only if it is sunny.
- It is not sunny this afternoon and it is colder than yesterday.
- If we do not go swimming, then we will take a canoe trip.
- If we take a canoe trip, then we will be home by sunset.
- Therefor (conclusion), we will be home by the sunset.

Prove the above argument is valid using inference rules by answering the following questions.

1. Translate the above statements into propositional logic [2 marks],

Write your answer here:

This question is literally asking you to write the above as propositional logic, so you just need to assign each statement a letter:
S: "it is sunny this afternoon"
C: "it is colder than yesterday"
W: "we will go swimming"
T: "we will take a canoe trip"
H: "we will be home by the sunset.

2. Write down the premises and the goal into propositional logic [2 marks],

Write your answer here:

Premises are declarative sentences, the statements that tell us the facts.
Goals are the statements to prove true or false.

Premises are:
- It is not sunny this afternoon and it is colder than yesterday. ¬S^C
- We will go swimming only if it is sunny. W → S
- If we do not go swimming, then we will take a canoe trip. ¬W → T
- If we take a canoe trip, then we will be home by sunset. T → H
Goal is:
- We will be home by the sunset. H

3. Write a formal proof, a sequence of steps that state premises or apply inference rules to previous steps. Justify each step by citing the rule of inference needed [6 marks].

Write your answer here:

A proof procedure is a method of proving statements using inference rules:

1. *modus ponens: if A→B and A are true, infer B is true,*

2. *modus tollens: if A→B and ¬A are true, infer ¬B is true,*

3. *elimination: if both (A ∧ B) is true, then infer both A and B are true,*

4. *introduction: if both A and B are true, then infer (A ∧ B),*

5. *disjunctive syllogism: if (A ∨ B) and ¬A are true, then infer B is true,*

Step	Reason
¬S ^ C	Premise
¬S	Simplification
W → S	Premise
¬W	Modus tollens of ¬S and W → S (S is not true, therefore W is not true)
¬W → T	Premise
T	Modus ponens of ¬W and ¬W → T (W is not true, therefore T is true)
T → H	Premise
H	Modus ponens of T and T → H (T is true therefore H is true)

Question 2: Artificial Neural Networks (ANNs) [20 marks in total]

a) Describe the basic model of an artificial neuron. (5 marks)

**Write your answer Here:**

Artificial neurons are often called nodes or units. It receives input. Each input has an associated weight. The unit computes an activation function of the weighted sum of its inputs.

b) A simple perceptron has two input units, a unipolar step function, weights $w_1 = 0.2$ and $w_2 = -0.5$, and a threshold $\theta = -0.2$ (θ can therefore be considered as a weight for an extra input which is always set to -1). What is the actual output of this perceptron for the input pattern $\mathbf{x} = [1, 1]^T$? (6 marks)

Write your answer Here:

Use the general model and step function ($y = 0$ if $x < 0$, $y = 1$ if $x > 1$), given at the end of the paper:

$$\mathbf{I} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad O = f\left(\sum_{j=1}^{n} w_j I_j - \theta\right) = f(\mathbf{w}^T \mathbf{I} - \theta) = f(net)$$
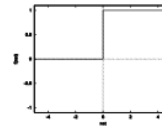
net $= (x_1 * w_1) + (x_2 * w_2) + (-1 * \vartheta)$
        $= (1 * 0.2) + (1 * -0.5) + (-1 * 0.2)$
        $= 0.2 \quad\quad - 0.5 \quad\quad + 0.2$
        $= -0.1$

y $= f(-0.1)$
       $= 0$

c) The previous perceptron is trained using the learning rule $\Delta \mathbf{w} = \eta (d - y) \mathbf{x}$, where $\mathbf{x}$ is the input vector, η is the learning rate, $\mathbf{w}$ is the weight vector, d is the desired output, and y is the actual output. What are the new values of the weights and threshold after one step of training with the input vector $\mathbf{x} = [0, 1]^T$ and desired output 1, using a learning rate $\eta = 0.2$? (9 marks)

Write your answer Here:

Recalculate the y value with the new x values (0 and 1). Following the equation from the previous question, we get $y = 0$. $\eta = 0.2$, $d = 1$, $y = 0$, $x = [0, 1]$ (so $x_1 = 0$ and $x_2 = 1$):
$\Delta w_1 = 0.2 * (1 - 0) * 0 = 0$
$\Delta w_2 = 0.1 * (1 - 0) * 1 = 0.2$

Part b says "ϑ can therefore be considered as a weight for an extra input which is always set to -1":
$\Delta w_3 = \Delta \vartheta = 0.2 * (1 - 0) * -1 = -0.2$

Δ means 'difference' so to get to the new w_x, we need to add the difference to the old w_x:
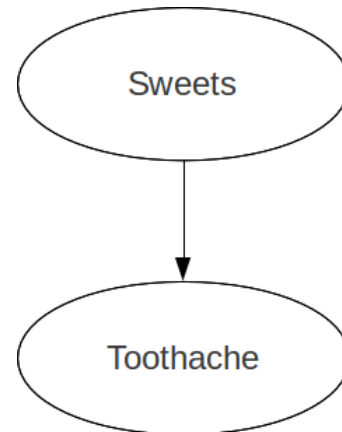
w_1 = 0.2 + 0 = 0.2
w_2 = -0.5 + 0.2 = -0.3
ϑ = -0.2 - 0.2 = -0.4

Question 3: Probabilistic Reasoning [20 marks in total]

Probabilistic Reasoning (medical expert system):

The following diagram shows a Bayesian belief network representing what happens when you eat sweets, i.e, if you eat sweets you may have toothache.

The conditional probabilities are (where S=sweets and T=toothache):

- P(S)=0.6   probability of eating sweets
- P(T|S)=0.8   indicates the probability of having toothache after eating sweets
- P(T|¬S)=0.1   indicates probability of having toothache without eating sweets.

Please answer the following questions, and make sure you evidence your way of thinking by using the correct formulas:

1. Calculate the prior probability P(¬S).   [3 marks]

Write your answer here:

The probability of not having sweets is the other option to having sweets:
P(¬S) = 1 – P(S) = 1 – 0.6 = 0.4

2. Calculate the conditional probability p(¬T|¬S).   [5 marks]

Write your answer here:

Not having toothache after not having sweets is the other option to having toothache after having sweets:
P(¬T|¬S) = 1 - P(T|¬S) = 0.9

3. Calculate P(T), i.e. the probability of having toothache   [6 marks]

Write your answer here:

To find the probability of having toothache, we need to add up the chances of all the possible ways you could have it. The probability of having toothache, if you've had sweets is 0.8, but the probability of having sweets in the first place is 0.6. Therefore, the overall probability of having toothache and having had sweets is as follows:
*P(T ^ S) = P(T|S) * P(S) = 0.8 * 0.6 = 0.48*
The probability of having toothache, if you've not had sweets is 0.1, but the probability of not having sweets in the first place is 0.4. Therefore, the overall probability of having toothache and having had sweets is as follows:
*P(T ^ ¬S) = P(T|¬S) * P(¬S) = 0.1 * 0.4 = 0.48*

Adding the two probabilities together gives the chance of having toothache overall:

<mark>*P(T) = P(T ^ S) + P(T ^ ¬S) = 0.48 + 0.04 = 0.52*</mark>

4. Calculate the conditional probability P(S|T), i.e. which is the probability of having eaten sweets if you have toothache [6 marks]

Write your answer here:

In this question, we will be using Bayes' Theorem, given at the bottom of the paper.

First, we'll swap the standard A and B for the relevant S and T, then add our values:

<mark>*P(S|T) = (P(T|S) * P(S)) / P(T)*
*= (0.8 * 0.6) / 0.52*
= 0.92</mark>

Question 4: Planning [20 marks in total]

Look a simple version of the "blocksworld" problem:
The objective of the "blocksworld" puzzle is to move objects by a robotic manipulator following the following constraints:

- The manipulator can only pick up one object at a time.
- Objects can either be put in one of three empty places on a table or on top of any other object that is free, i.e., has not yet an object on top of it.
- Only objects that have nothing stacked on top of them can ever be picked up.

1. The initial state is depicted below. It shows two objects "b_1" and "b_2" stacked in place 1. Describe this initial state completely in predicate logic (in a syntax similar to PDDL), using the predicate (on ?a ?b) and (free ?a).   [5 marks]

Write your answer here:

This question is asking us to describe the diagram above, using functions and constants in statements in the style of PDDL:
(on b_2 b_1)
(on b_1 place_1)
(free b_2)
(free place_2)
(free place_3)

2. Now look at the goal state below. Again, use predicate logic to describe the *relevant* constraints of this goal state using the same predicated as above:

[3 marks]

Write your answer here:

"Relevant constraints" means only the differences between the initial and goal states:
(on b_1 b_2)
(on b_2 place_3)

3. Define an action "(move ?object ?from ?to)" that takes a free object "?object" from either an object or place "?from" and puts it onto either a place or other object "?to" that is also initially free. Describe the precondition and the effect of your action in predicate logic (PDDL-like syntax).   [7 marks]

Write your answer here:

If the preconditions are met, the effects are performed. If the object is free (it has no other object on it), if the position to move to is free, and if the object is on its position to be moved from, set the destination position to not free, make it so that the object is not on its previous position, but is on its destination, and set the previous position to free:
(move ?object ?from ?to):
 preconditions:
   (free ?object)
   (free ?to)
   (on ?object ?from)
 effects:
   (not (free ?to))
   (not (on ?object ?from))
   (on ?object ?to)
   (free ?from?)

4. Draw a search tree (up to depth 3, i.e. two move actions to be executed) using the predicate notation above, i.e., in each node of the search tree indicate the predicates that are true, and on each edge describe the action that is executed. The outcome should look similar to this tree (obviously, you need to complete the nodes and also the number of leaves will be different for you): [5 marks]
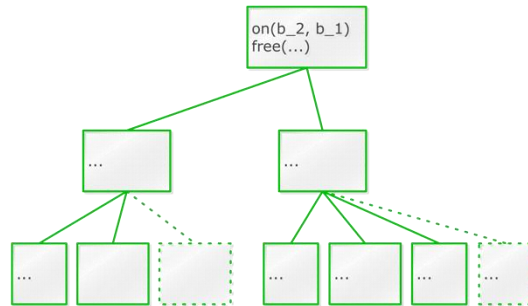
Write your answer here:

I think this is right, the mark scheme is vague, but I have emailed Miao.
Search trees graph every possibility to find the shortest route to the goal. Using the move action in the situation above, there are two initial options. Either move b_1 to place_2 or to place_3.
From there, we can either move b_1 to b_2, move b_1 to place_2, move b_2 to place_2, or move b_2 to b_1 (the goal). The search tree stops looking as soon as it gets to the goal state, so we can draw the tree as if it tried moving b_2 to b_1 as its first third-level action.
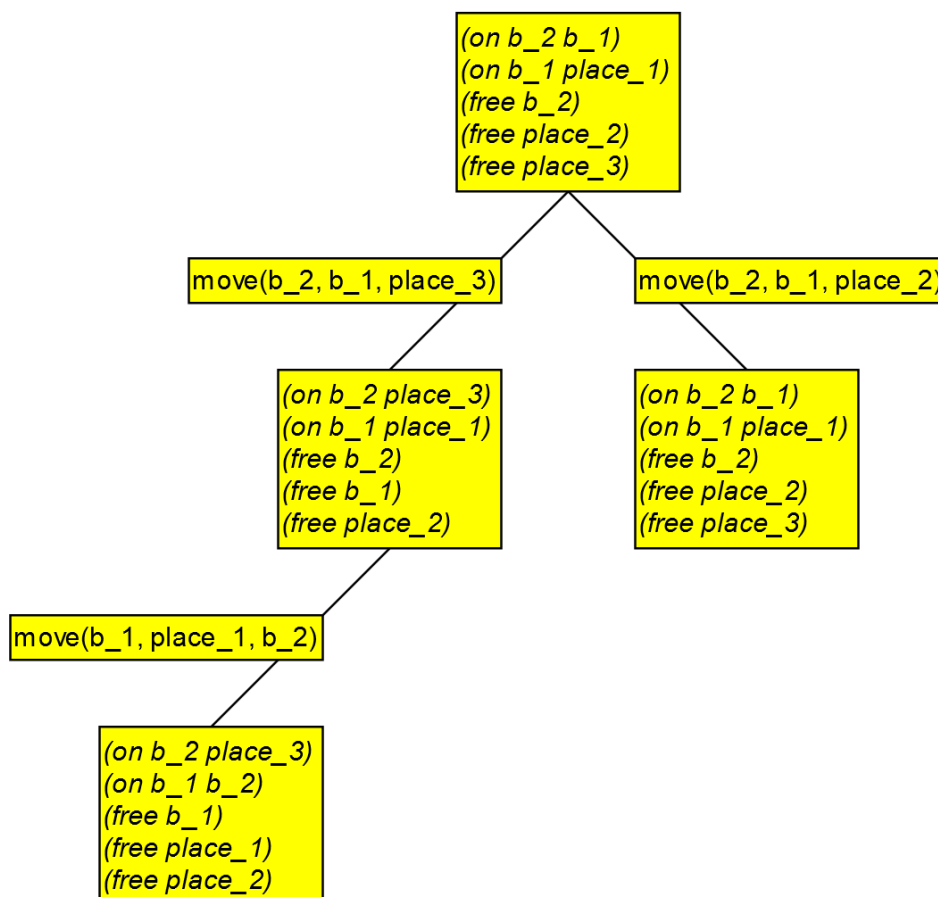
Question 5: Search [20 marks in total]

1. Using the following node tree, explain the difference between depth-first and breadth-first search. What is the node search order in each case? (6 marks)
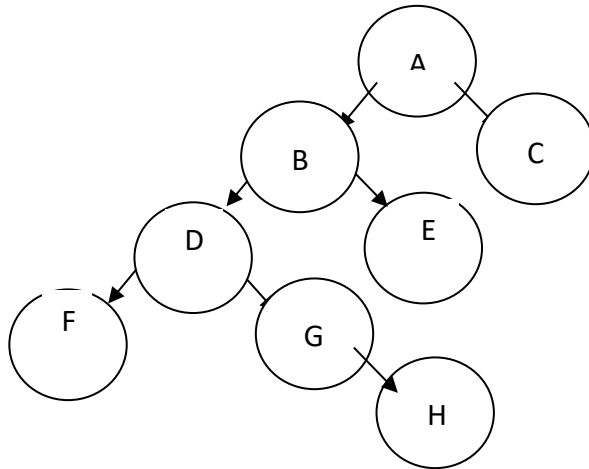
Write your answer here:

Depth-first searches down the tree in each way possible: ABDFGHEC.
Breadth-first searches each layer, starting at the root: ABCDEFGH.

2. Explain how Dijkstra's algorithm is initialised for a grid-based world, with a known start location. (4 marks)

Write your answer here:

Set every node's total cost to infinity, except for the start node which should be set to zero. All nodes are put into the open list and their links are set arbitrarily (usually to null).

3. What are the "open" and "closed" lists used for in Dijkstra's algorithm? (2 marks)

Write your answer here:

*The open list contains nodes whose total costs may still be evaluated and updated.
The closed list contains nodes whose costs and links have been finalised.*

4. In Dijkstra's algorithm, how is a pair-cost used to update the total cost for a node after its neighbour node is closed? (4 marks)

Write your answer here:

The total cost for the node is re-estimated as the total cost for the closed neighbour plus the cost from that neighbour to the node and the node's link is updated to the neighbour, if the current node total cost is higher than the new estimate and the node isn't already closed.

5. Give an example of a specific heuristic distance function for A* that could be used in a grid-based world. (2 marks)

Write your answer here:

> *Euclidean, Chebyshev, or Manhattan.*

6. What does the term "admissible heuristic" mean in relation to the A* algorithm? (2 marks).

Write your answer here:

> *It does not over-estimate the actual shortest distance from the node to the target.*

Question 6: Games AI [10 marks in total]

1. State two fundamental requirements of an AI system for a video game (2 marks).

Write your answer here:

Speed, predictability, low resource requirements, believability.

2. Describe an extension to the A* algorithm which uses a 3D grid to plan non-colliding 2D paths for multiple agents moving simultaneously through the same map. (5 marks)

Write your answer here:

Third dimension represents time. Agents move from layer to layer through the time dimension at each update step. Agents should wait for others to find their path so that their paths can be added to the 3D map. Agent paths can be processed as normal.

3. What is path-patching in A*, and when might it be used? (3 marks).

Write your answer here:

Path-patching is the act of recalculating a sub-section of an existing path if the path becomes blocked by another object or a change to the environment.

Appendix: Useful Formulas

Truth Tables

operator ¬ Negation (not)	
P	**¬P**
T	F
F	T

operator ∧ Conjunction (and)

P	Q	P∧Q
T	T	T
T	F	F
F	T	F
F	F	F

operator ∨ Disjunction (or)

P	Q	P∨Q
T	T	T
T	F	T
F	T	T
F	F	F

operator → Implication

P	Q	P→Q
T	T	T
T	F	F
F	T	T
F	F	T

operator ≡ Equivalence

P	Q	P≡Q
T	T	T
T	F	F
F	T	F
F	F	T

Inference Rules

- Modus ponens: If P and P → Q are true, then infer Q.
- Modus tollens: If P → Q and ¬Q are true, then infer ¬P.
- And elimination: If P ∧ Q is true, then infer both P and Q are true
- And introduction: If both P and Q are true, then infer P ∧ Q
- Disjunctive syllogism (DS): If addition (P ∨ Q) and ¬P are true, then infer Q

Probability Theory

Mathematically, conditional probability is defined as:
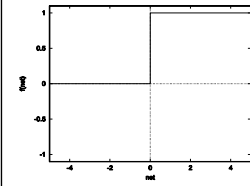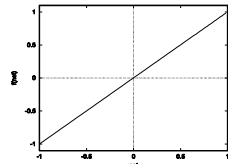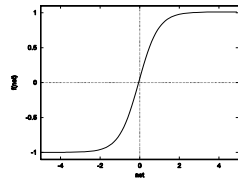
$$P(A \mid B) = P(A \wedge B) / P(B)$$

Note that $P(B) \neq 0$.

Bayes Theorem (general)

$$P(H \mid E) = \frac{P(H)P(E \mid H)}{P(E)}$$

Artificial Neuron

General Model	
$\bar{\mathbf{I}} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix}$ $\bar{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$	$O = f\left(\sum_{j=1}^{n} w_j I_j - \theta \right) = f\left(\bar{\mathbf{w}}^T \bar{\mathbf{I}} - \theta \right) = f(net)$

Activation Functions

Step:	$f(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases}$	Linear:	$f(net) = net$	Sigmoid:	$f(net) = \dfrac{2}{1 + e^{-\lambda net}} - 1$

Gradient Descent Algorithm

GRADIENT-DESCENT($training\_examples, \eta$)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o
    - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$