NATIONAL UNIVERSITY OF SINGAPORE

IT5005 – ARTIFICIAL INTELLIGENCE

(Semester 1: AY2023/24)

Time Allowed: 2 Hours

INSTRUCTIONS

1. This assessment paper contains **FIFTEEN (15)** questions in **TWO (2)** parts and comprises **EIGHT (08)** printed pages.
2. This is a **CLOSED BOOK** assessment.
3. Only an A4 cheat sheet is allowed. Apart from calculators, electronic devices are not allowed.
4. Answer **ALL** questions and write your answers only on the **ANSWER SHEET** provided.
5. Do **not** write your name on the ANSWER SHEET. Please write your Student Number only.
6. The maximum mark of this assessment is 100.

Question	Max. mark
Part A: Q1 – 10	20
Part B: Q11 (Propositional Logic)	25
Part B: Q12 (First Order Logic)	20
Part B: Q13 (Bayesian Network)	15
Part B: Q14 (MDP)	10
Part B: Q15 (Neural Networks)	10
Total	**100**

——— END OF INSTRUCTIONS ———

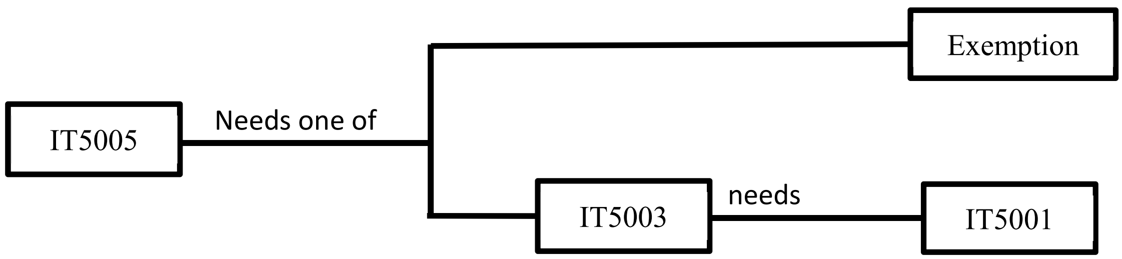
Part B: There are 5 questions in this part [Total: 80 marks]

11. [Total: 25 marks] Propositional Logic.

(a) Prove that the following statement is tautology. You should NOT use truth-table enumeration for the proof.

$(\neg P \vee \neg Q \vee R) \wedge (\neg R \vee S \vee T) \Rightarrow (\neg P \vee \neg Q \vee S \vee T)$ [5 marks]

(b) A special master's program of NUS SoC requires the students to pass the essential modules IT5001, IT5003, and IT5005 to graduate. Prerequisite tree for these modules is shown below:



The prerequisite tree tells us that IT5001 is a prerequisite for IT5003, i.e., students cannot register for IT5003 unless they register and pass the module IT5001. Similarly, the students cannot register for IT5005 unless they register and pass the module IT5003. Few students get exemption for IT5001 and IT5003. Otherwise, the students must pass IT5001 and IT5003 to register for IT5005. We use the following propositional symbols:

R1: Aiken register for IT5001	R3: Aiken register for IT5003	R5: Aiken register for IT5005
P1: Aiken pass IT5001	P3: Aiken pass IT5003	P5: Aiken pass IT5005
E: Aiken is exempted from IT5001 and IT5003		

(i) Convert the following sentences into propositional sentences. [10 marks]

1. Unless Aiken register for IT5001, Aiken does not pass IT5001.
2. Unless Aiken register for IT5003, Aiken does not pass IT5003.
3. Unless Aiken register for IT5005, Aiken does not pass IT5005.
4. Aiken register for IT5003 only if Aiken pass IT5001.
5. Aiken register for IT5005 only if either Aiken pass IT5003 or Aiken is exempted from IT5001 and IT5003.

(ii) You are given the following facts:

Fact 1: Aiken pass IT5005.

Fact 2: Aiken is not exempted from IT5001 and IT5003

You are given the following queries: "Aiken pass IT5001" and "Aiken pass IT5003"

Which algorithm would you recommend to prove this query? Explain the rationale and prove the query using the selected algorithm. [10 marks]

12. [Total: 20 marks] FOL

(a) Find the most general unifier for the predicates [10 marks]

$Q(z, G(f(A), y))$ and $Q(G(x, f(B)), z)$

(b) Consider the following argument: [10 marks]

No college cafeteria food is good

No good food is wasted

∴ No college cafeteria food is wasted

Is this argument valid? You need to justify your reasoning using the following predicates.

$C(x)$: food x is college cafeteria food

$G(x)$: food x is good

$W(x)$: food x is wasted

You may assume that the domain of discourse is X and it represents the set of all foods.

13. [Total: 15 marks] Bayesian Networks.

Consider a Bayesian network with four propositional variables W, X, Y, and Z shown below:

W	P(X=0|W)
0	0.9
1	0

X	Y	P(Z=0|X, Y)
0	0	0.9
0	1	0.2
1	0	0.1
1	1	0

The values 0 and 1 indicate that the corresponding variables are false and true, respectively. The variables W and Y take the value 0 with probabilities of 0.9 and 0.6, respectively. The conditional probability tables of the variables X and Z are shown above.

(a) Compute $P(W = 1, X = 1, Y = 1, Z = 1)$. [3 marks]

(b) What is the probability that $X = 1$? [4 marks]

(c) What is the probability that $X = 1$ given $Y = 1$? [4 marks]

(d) What is the probability that $W = 1$ given $X = 1$? [4 marks]

14. [Total: 10 marks] Markov Decision Processes.

Consider an MDP with three states S_1, S_2 and S_3. State S_3 is the terminal state and the agent collects a reward of -10 for the transition towards terminal state. The agent can take two actions a_1 and $a_2$in non-terminal states. The agent collects a reward of +1 with action a_1 in state S_1 and stays in the same state. However, with action a_2 in state S_1, the agent stays in the same state with probability 0.5 with a reward of +2 and moves to state S_2 with probability 0.5 and reward of +2. In state S_2, action a_1 takes the agent to either state S_1 or S_2 with probability 0.5 and reward +1. Action a_2 in state S_2 takes the agent to terminal state with a reward of -10. Assume a discount factor of 0.8.

(a) Find the utilities of all states with policy π, where $\pi(S_1) = \pi(S_2) = a_1$. [6 marks]

(b) Write the Bellman update equations for utilities of states S_1 and S_2 [4 marks]

15. [Total: 10 marks] Neural Networks.

(a) Grace has designed a single layer neural network with single input variable x and single output variable $\hat{y}$. The output of the neural network is modeled as $\hat{y} = g(f(x))$, where $f(x) = wx + b$ and $g(x) = \frac{1}{1+e^{-x}}$ is the activation function. Furthermore, the weight (w) and bias (b) parameters are initialized as 1 and 0, respectively. Find $\frac{\partial \hat{y}}{\partial w}$ at $x = 0$ and $x = 1$? [4 marks]

(b) Aiken argued that adding hidden layers improve the prediction performance. The input (x), weights (w_1 and w_2), and bias ($b_1$and b_2) are scalars. Aiken revised the model such that the output of the predictor is $\hat{y} = g^{[2]}(f^{[2]}(g^{[1]}(f^{[1]}(x))))$, where $f^{[i]}(x) = w_i x + b_i$, $i = \{1,2\}$. Moreover, $g^{[i]}(x) = \tanh(x)$, $i = \{1,2\}$, are selected as activation functions. Find $\frac{\partial \hat{y}}{\partial w_1}$ and $\frac{\partial \hat{y}}{\partial w_2}$ at $x = 1$? [6 marks]

You may use the following fact: $\frac{d(\tanh(x))}{dx} = 1 - (\tanh(x))^2$

Appendix:

Logical Equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Pseudo Code for Resolution Algorithm

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
      α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new ← { }
  while *true* **do**
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

Pseudo Code for Forward Chaining Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
     p ← POP(queue)
     if p = q then return true
     if inferred[p] = false then
        inferred[p] ← true
        for each clause c in KB where p is in c.PREMISE do
           decrement count[c]
           if count[c] = 0 then add c.CONCLUSION to queue
  return false
```

Pseudo Code for Unification Algorithm

```
function UNIFY(x, y, θ=empty) returns a substitution to make x and y identical, or failure
  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
     return UNIFY(ARGS(x), ARGS(y), UNIFY(OP(x), OP(y), θ))
  else if LIST?(x) and LIST?(y) then
     return UNIFY(REST(x), REST(y), UNIFY(FIRST(x), FIRST(y), θ))
  else return failure

function UNIFY-VAR(var, x, θ) returns a substitution
  if {var/val} ∈ θ for some val then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ for some val then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

=== END OF PAPER ===