NATIONAL UNIVERSITY OF SINGAPORE

CS3243 - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

(Semester 2: AY2014/15)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. Please write your Student Number (Matriculation Number) only. Do not write your name.
2. This assessment paper contains **FIVE (5)** parts and comprises **THIRTEEN (13)** printed pages, including this page.
3. Answer **ALL** questions as indicated.
4. This is a **OPEN BOOK** assessment.
5. You are allowed to use **NUS APPROVED CALCULATORS**.
6. Please write your **Matriculation Number** below.

MATRICULATION NUMBER: ____________________

EXAMINER'S USE ONLY		
Part	Mark	Score
I	5	
II	13	
III	6	
IV	10	
V	16	
TOTAL	50	

In Part I, II, III, IV, and V, you will find a series of short essay questions. For each short essay question, give your answer in the reserved space in the script.

Part I
Inference in First-Order Logic

(5 points) Short essay questions. Answer in the space provided on the script.

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound expression
          y, a variable, constant, list, or compound expression
          θ, the substitution built up so far (optional, defaults to empty)

  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(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) and LIST?(y) then
      return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure

function UNIFY-VAR(var, x, θ) returns a substitution

  if {var/val} ∈ θ then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

Figure 1: Unification algorithm.

1. (4 points) Give the most general unifier (if it exists) for the pair of sentences $V(U(W, z), y, T(J))$ and $V(y, U(x, T(a)), z)$ that is **produced** by the unification algorithm given in Fig. 9.1 of AIMA 3rd edition (reproduced in Figure 1) where W and J are constants, V is a predicate, U and T are functions, and x, y, z, and a are variables. You don't have to show your derivation.

Solution:

2. (1 point) We know that the most general unifier (MGU) is unique up to renaming and substitution of variable(s). Give one such MGU (if it exists) that is equivalent to that in the above question by **substitution of variable(s)**. You don't have to show your derivation.

Solution:

Part II
Adversarial Search

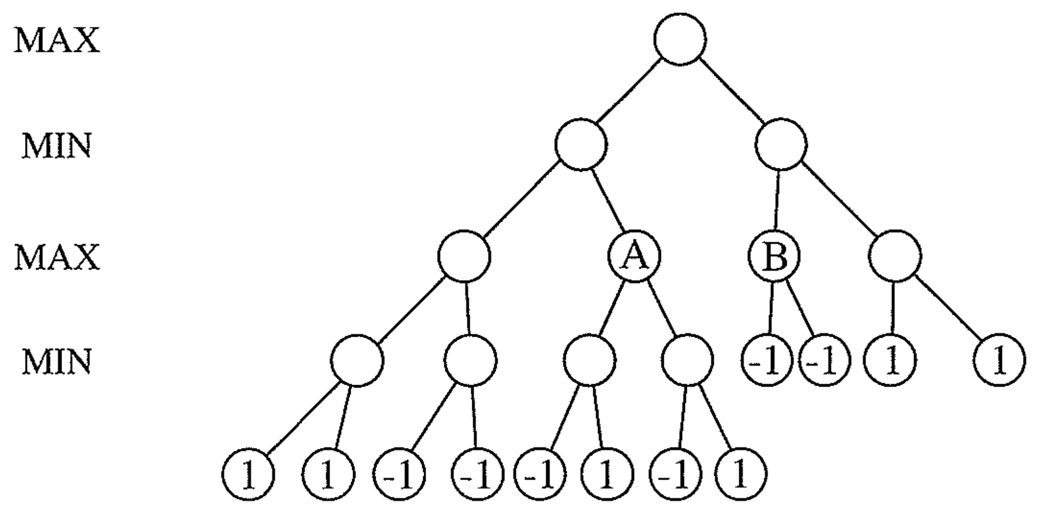
(13 points) Short essay questions. Answer in the space provided on the script.

1. (9 points) Consider the minimax search tree shown in the solution space below; the utility function values specified with respect to the MAX player and indicated at all the leaf (terminal) nodes. Suppose we use alpha-beta pruning algorithm, given in Figure 5.7 of AIMA 3rd edition (reproduced in Figure 2), in the direction **FROM LEFT TO RIGHT** to prune the search tree. **Mark (with an "X") all ARCS** that are pruned by alpha-beta pruning.

```
function ALPHA-BETA-SEARCH(state) returns an action
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in ACTIONS(state) with value v
--------------------------------------------------------
function MAX-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v
--------------------------------------------------------
function MIN-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

Figure 2: Alpha-beta pruning algorithm.

Solution:



When the alpha-beta pruning algorithm **first** visits MAX node 'A' by calling the MAX-VALUE function, what is the **exact** value of β?

Solution: $\beta =$

When the alpha-beta pruning algorithm **first** visits MAX node 'B' by calling the MAX-VALUE function, what is the **exact** value of α?

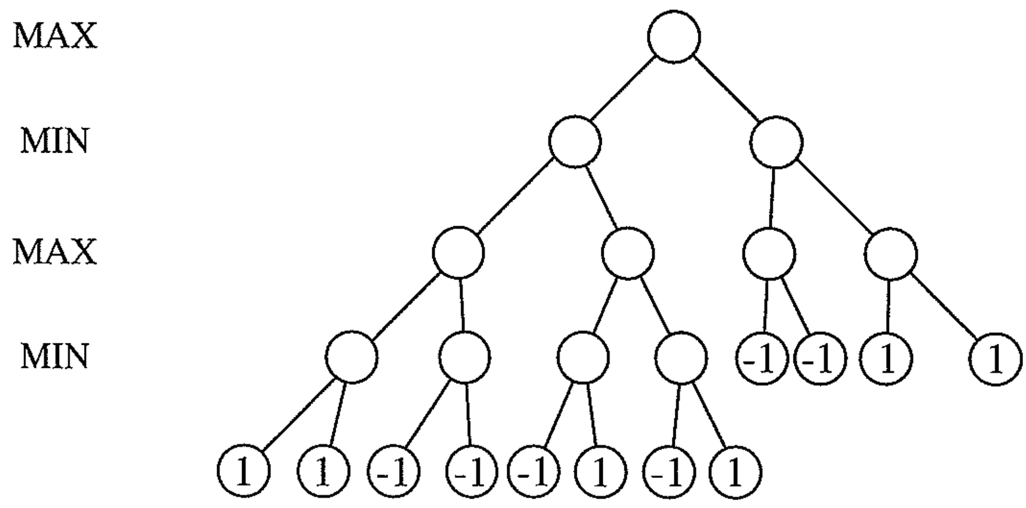
Solution: $\alpha =$

State the **EXACT** minimax value at the root node.

Solution:

2. (3 points) For the same minimax search tree shown in the solution space below, observe that the utility function values specified with respect to the MAX player and indicated at all the leaf (terminal) nodes are either -1 or 1. This observation can be exploited to modify the alpha-beta pruning algorithm for improving the search efficiency while the search tree is being pruned in the direction **FROM LEFT TO RIGHT**. To achieve this, the line of code '**if** $v \geq \beta$ **then return** v' in the MAX-VALUE function is modified to '**if** $(v = 1 \vee v \geq \beta)$ **then return** v'. **Mark (with an "X") all ARCS** in the minimax search tree in the solution space below that are pruned by alpha-beta pruning utilizing this line of modified code.

Solution:



3. (1 point) Using a **similar** trick as in the question above (i.e., by exploiting the observation described in the question above), state exactly **ONE** line of code to modify '**if** $v \leq \alpha$ **then return** v' in the MIN-VALUE function to improve the search efficiency of alpha-beta pruning.

Solution:

Part III
Uncertainty

(6 points) Short essay questions. Answer in the space provided on the script.

A unique trait of Singapore is its culture of tuition/enrichment classes for the children (apart from attending school). Bryan is having doubts about whether to continue sending his 4-year-old daughter, Cara, to her current tuition/enrichment classes because he is worried that tuition may adversely affect her passion for learning. Bryan has some initial prior belief about whether Cara has great or mild passion for learning:

$$\mathrm{P}(Passion = great) = 0.7, \qquad \mathrm{P}(Passion = mild) = 0.3\,.$$

During times when Cara is greatly passionate about learning, Bryan observes Cara's preferences for excessive tuition, some tuition, and no tuition with the following respective probabilities:

$$\mathrm{P}(excessivetuition \mid great) = 0.1, \quad \mathrm{P}(sometuition \mid great) = 0.5, \quad \mathrm{P}(notuition \mid great) = 0.4\,.$$

On the other hand, during moments when Cara is mildly interested in learning, Bryan notices Cara's preferences for excessive tuition, some tuition, and no tuition with the following respective probabilities:

$$\mathrm{P}(excessivetuition \mid mild) = 0.0, \quad \mathrm{P}(sometuition \mid mild) = 0.2, \quad \mathrm{P}(notuition \mid mild) = 0.8\,.$$

1. (4 points) Suppose that, at this time, Bryan observes Cara's preference for no tuition. Given this observation, calculate the posterior belief that Cara is mildly interested in learning. Show your derivation. No marks will be given if you do not show your derivation. Give your answers up to 5 decimal places.

Solution:

$\mathrm{P}(mild \mid notuition) =$

2. (2 points) Bryan likes to be more certain about whether Cara has indeed become mildly interested in learning. Specifically, Bryan wants to be at least 75% certain that Cara is mildly interested in learning. In question 1, it is noted that Bryan has already observed that Cara prefers no tuition on the first day. Suppose that Bryan has consecutively observed that Cara prefers no tuition on **each day** for the **next** C **days**. Given these observations, state the minimum value of C in order to achieve at least 75% certainty that Cara is mildly interested in learning. You can assume conditional independence between different observations given Cara's passion for learning (i.e., either great or mild).

Solution:

$C =$

Part IV
Learning from Examples

(10 points) Short essay questions. Answer in the space provided on the script.

In this question, we will build a decision tree using the 8-example training set (see Table 1) to understand the factors affecting the graduation of students enrolled in our computer science Ph.D. program. This scenario is purely hypothetical.

Example	Input Attributes			Goal
Ph.D. Student	*Supervisor*	*Motivation*	*Sociable*	*Graduate?*
Jaemin	*Mean*	*High*	*No*	*Yes*
Etkin	*Mean*	*Average*	*Yes*	*No*
Hancheng	*Kind*	*Poor*	*No*	*No*
XuNuo	*Kind*	*Average*	*Yes*	*Yes*
ChunKai	*Mean*	*High*	*No*	*Yes*
QuocPhong	*Kind*	*Average*	*Yes*	*Yes*
Yehong	*Kind*	*High*	*Yes*	*Yes*
Ruofei	*Mean*	*Poor*	*No*	*No*

Table 1: Examples for the *AIGrade* domain.

1. (3 points) What is the entropy of the goal attribute '*Graduate?*' on the whole set of examples shown in Table 1? Give your answer up to 4 decimal places.

Solution:

$H(Goal) =$

What is the information gain with choosing the input attribute '*Supervisor*' as the root of the decision tree? Give your answer up to 4 decimal places.

Solution:

$Gain(Supervisor) =$

What is the information gain with choosing the input attribute '$Motivation$' as the root of the decision tree? Give your answer up to 4 decimal places.

Solution:

$Gain(Motivation) =$

2. (5 points) Using the DECISION-TREE-LEARNING algorithm given in Fig. 18.5 of AIMA 3rd edition (reproduced in Figure 3) and information gain as the IMPORTANCE function in this algorithm, draw the resulting decision tree that is induced by the 8-example training set. You are required to label the non-leaf nodes with the input attributes, the leaf nodes with the values of the goal attribute '$Graduate?$' (i.e. either 'Yes' or 'No'), and the branches with the values of the chosen attributes.

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** tree

  if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
  else
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
    tree ← a new decision tree with root test A
    for each value v_k of A **do**
      $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
      subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* − A, *examples*)
      add a branch to *tree* with label ($A = v_k$) and subtree *subtree*
    return *tree*

Figure 3: Decision tree learning algorithm.

Solution:

3. (2 points) Give the logical expression (in disjunctive normal form) that corresponds to the decision tree produced by the DECISION-TREE-LEARNING algorithm. Let 'Yes' and 'No' be associated with True and False, respectively.

Solution:

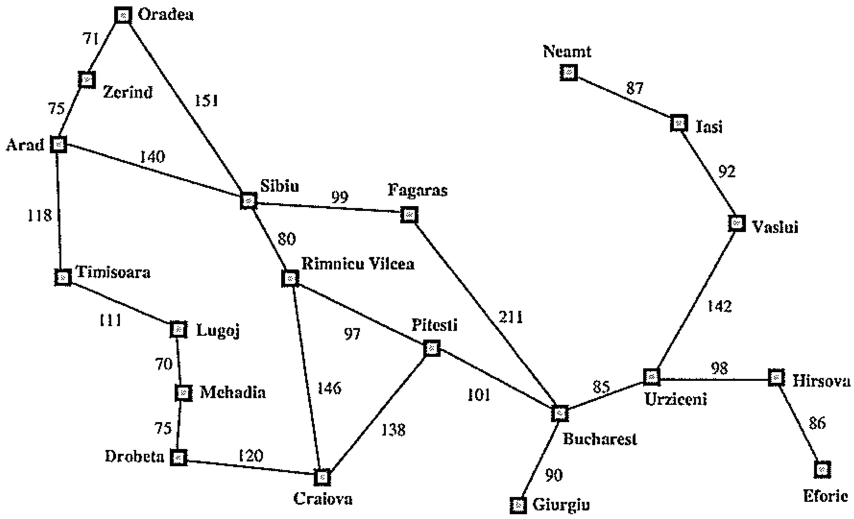
Part V
Informed Search

(16 points) Short essay questions. Answer in the space provided on the script.

Consider the graph in Figure 4 below for **ALL** the questions in Part V. Apply the graph search algorithms indicated below to find a path from **BUCHAREST** to **SIBIU** using the heuristic function (if necessary)

$$h(n) = \max\left(h_{SLD}(\text{Sibiu}) - h_{SLD}(n), \frac{h_{SLD}(n) - h_{SLD}(\text{Sibiu})}{4} \right)$$

where $h_{SLD}(n)$ is the straight-line distance from any city n to Bucharest given in Figure 3.22 of AIMA 3rd edition (reproduced in Figure 4).



Values of h_{SLD} - straight-line distances to Bucharest

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 4: Graph of Romania.

1. (6 points) Trace the **A* search algorithm using GRAPH SEARCH** and the evaluation function $f(n) = g(n) + h(n)$ by showing the nodes in the frontier at the end of each iteration of the outer loop. **Pay very careful attention to the following instructions when presenting your solution:**

   - Recall from page 93 of AIMA 3rd edition (specifically, last line of text) that the A* search algorithm is identical to uniform-cost search (reproduced from Figure 3.14 of AIMA 3rd edition in Figure 5 below) except that A* uses $f = g + h$ instead of g.
   - For each node n in the frontier, give the corresponding 3-tuple $(g(n), h(n), f(n))$.
   - At the end of each iteration of the outer loop, list the nodes in the frontier in nondecreasing order of f value.
   - AFTER the goal node is found (i.e., last iteration of the outer loop), you must also list the nodes in the frontier.
   - If **tie-breaking** is needed between two nodes in the frontier with the same lowest f value, then expand the node with the smaller h value first.

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
      if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              frontier ← INSERT(child, frontier)
          else if child.STATE is in frontier with higher PATH-COST then
              replace that frontier node with child
```

Figure 5: Uniform-cost search algorithm.

Solution: The node (denoting the initial state) in the frontier before entering the outer loop is provided.

FRONTIER:

Bucharest(0,253,253)
End of Iteration 1:
End of Iteration 2:
End of Iteration 3:
End of Iteration 4:
End of Iteration 5:
End of Iteration 6:

2. (1 point) Give the solution path from Bucharest to Sibiu that is produced by the A* search algorithm.

Solution:

3. (4 points) Prove that $h(n)$ is a consistent heuristic (*Hint*: Consider using the triangle inequality).

Solution:

4. (2 points) Let $h_1(n) = \max\left(h_{SLD}(\text{Sibiu}) - h_{SLD}(n), \dfrac{h_{SLD}(n) + h_{SLD}(\text{Sibiu})}{4}\right)$. Prove that $h_1(n)$ is NOT an admissible heuristic for A* tree search to determine an optimal path from Bucharest to Sibiu.

Solution:

5. (1 point) Give the solution path from Bucharest to Sibiu that is produced by the uniform-cost search algorithm given in Figure 3.14 of AIMA 3rd edition (reproduced in Figure 5).

Solution:

6. (2 points) We have claimed during lecture that the uniform-cost search algorithm given in Figure 3.14 of AIMA 3rd edition (reproduced in Figure 5) is optimal. Explain concisely (i.e., in at most 3 sentences) why uniform-cost search is optimal using the result that "If a heuristic is consistent, then A* graph search is optimal". No points will be awarded if such a result is not being used.

Solution:

END OF PAPER