

EXAMINATIONS — 2004

END-YEAR

COMP 307
ARTIFICIAL INTELLIGENCE
Sample Answers

Time Allowed: 3 Hours

Instructions: There are a total of 180 marks on this exam.
Attempt all questions.
Calculators may be used.
Non-electronic foreign language translation dictionaries may be used.

Questions

- | | |
|--------------------------------|------|
| 1. Prolog | [30] |
| 2. Search | [26] |
| 3. Rule Based Systems | [30] |
| 4. Planning | [35] |
| 5. Natural Language Processing | [23] |
| 6. Machine Learning | [36] |

Student ID:

Question 1. Prolog

[30 marks]

(a) [12 marks] Matching (Unification)

Give the most general unifier for the following pairs of terms, or say that the terms do not unify. The first answer is given as an example.

property(One, Two)
property(3, 4)

Unifies with One=3, Two=4

date(D, M, 2001)
date(15, M, Y)

Unifies with D=17, Y=2001

point(A, B)
point(X, Y, Z)

Does not unify

plus(2, 2)
4

Does not unify

2 + D
E + 2

Unifies with E=2, D=2

triangle(point(-1,0), P2, P3)
triangle(P1, point(1,0), point(0, Y))

Unifies with P1=point(-1,0), P2=point(1,0),
P3=point(0,Y)

lect(X, room(hu1), [comp307, X])
lect(Y, room(Y), Z)

Unifies with X=Y=hu1, Z=[comp307, hu1]

Student ID:

(b) [10 marks] How Prolog answers a question

Consider the following program.

```
p(a, b).  
p(a, c).  
p(c, d).  
p(c, e).  
  
q(f, c).  
q(e, g).  
  
r(X, Y) :- q(X, Y).  
r(X, Y) :- p(X, Y).  
  
s(X, Y) :- r(X, Z), r(Z, Y).
```

Write out the output generated by the following queries.

Give all the possible answers in the order they are generated.

(i) `|?- r(X, Y).`

```
| X = f, Y = c ? ;  
| X = e, Y = g ? ;  
| X = a, Y = b ? ;  
| X = a, Y = c ? ;  
| X = c, Y = d ? ;  
| X = c, Y = e ? ;  
| no
```

(ii) `|?-s(a, X).`

```
| X = d ? ;  
| X = e ? ;  
| no
```

Student ID:

(c) [8 marks] **Prolog Programming**

Define the relation `translate(List1, List2)` to translate a list of digits between 0 and 5 to a list of the corresponding words. For example,

```
translate([3, 5, 1, 3], X) is true with X=[three, five, one, three]
translate([2, 0], [two, zero]) is true
translate([ ], X) is true with X=[ ]
```

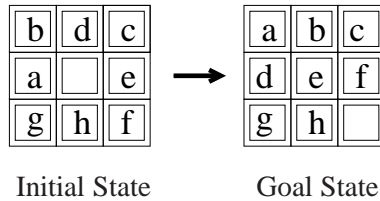
```
| translate([ ],[ ]).
| translate([D | Digits],[W | Words]) :- translate(D,W), translate(Digits, Words).

| translate(0, zero).
| translate(1, one).
| translate(2, two).
| translate(3, three).
| translate(4, four).
| translate(5, five).
```

Question 2. Search

[26 marks]

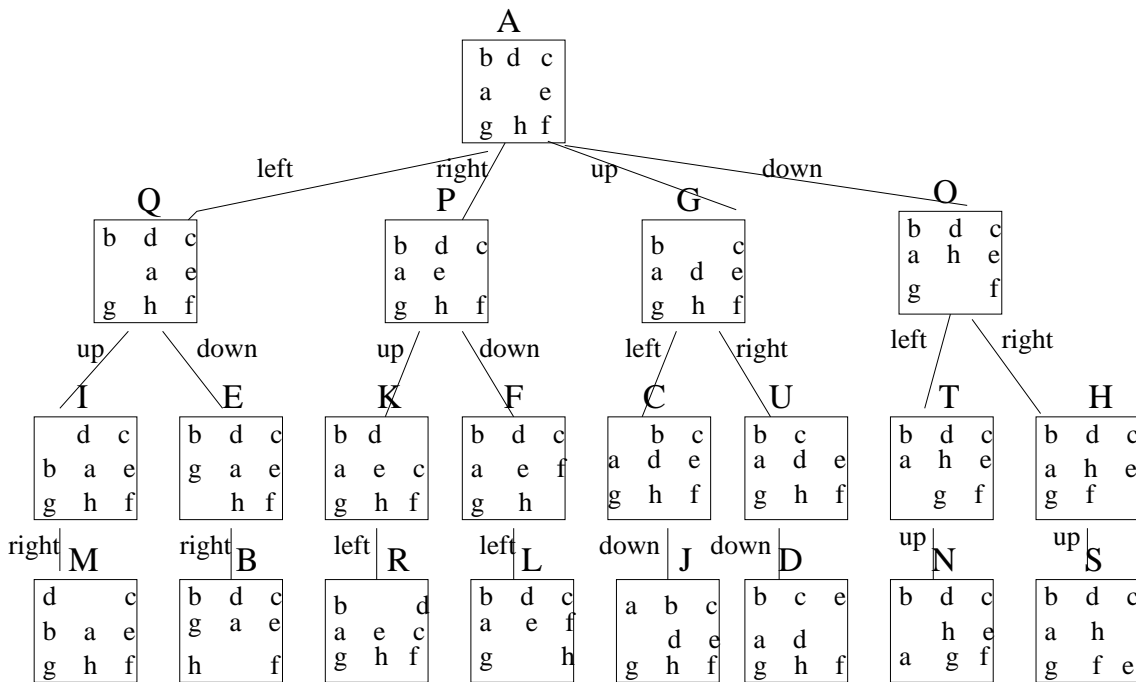
This question concerns a search based problem solver for the 8-puzzle problem. The initial state and the goal state are as follows:



Assume that the problem solver has a loop detection system to avoid visiting the same state more than once, and always considers the four operators in the following order:

- Move the blank to left
- Move the blank to right
- Move the blank up
- Move the blank down

The figure below shows part of the search tree that the problem solver must search.



(a) [6 marks] List the labels of the first 11 search tree nodes in the order in which the nodes will be expanded for each of the following uninformed search strategies. State any additional assumptions you made.

Breadth first search

A Q P G O I E K F C U

Student ID:

Depth first search

| A Q I M E B P E R F L

Iterative deepening search

| (A) A Q P G O A Q I E P (K)

(b) [10 marks] The informed search strategies such as Greedy search and A* search use heuristics to guide the search. Define a reasonable heuristic function for the 8-puzzle (in words, not Prolog) and compute the value of your function on the states Q, P, G, and O in the search tree.

| Number of tiles out of place, OR
| Sum of the Manhattan distances of each tile from the goal position

(c) [10 marks] Suppose the problem solver uses A* search with your heuristic from (b) to solve the problem. Using the search tree on the facing page, list the first four nodes to be expanded and give the value of every node that was evaluated.

|

Student ID:

Question 3. Rule Based Systems

[30 marks]

Consider the following set of rules.

rule1 :: **if** kitchen_dry and hall_wet **then** leak_in_bathroom

rule2 :: **if** hall_wet **and** bathroom_dry **then** problem_in_kitchen

rule3 :: **if** problem_in_kitchen **and** no_water_from_outside **then** leak_in_kitchen

rule4 :: **if** window_closed **then** no_water_from_outside

rule5 :: **if** no_rain **then** no_water_from_outside

leak_in_bathroom and leak_in_kitchen are goals to be established. kitchen_dry, hall_wet, bathroom_dry, window_closed, no_rain are the questions that can be asked of the user. Assume the rules are scanned top down, that is, a rule with a smaller number has higher priority.

(a) [6 marks] Backward chaining

Suppose a backward chaining interpreter uses these rules to reach a diagnosis. It will ask the user when a fact can be asked, and will stop with the first goal that it discovers to be true. The answers for hall_wet, bathroom_dry, no_rain are yes and answers for others are no. Assume that it does **not** cache answered questions, and asks again if necessary.

List the questions in the order that they are asked by the backward chaining interpreter.

kitchen_dry, hall_wet, bathroom_dry, window_closed, no_rain

(b) [6 marks] Forward chaining

Suppose a forward chaining interpreter uses these rules to reach a diagnosis. hall_wet, bathroom_dry, no_rain are true and are saved in the working memory. Assume that every time it adds a fact to the working memory, it scans the rules from the beginning again.

List the new facts in the order that they are added to the working memory.

problem_in_kitchen, no_rain_from_outside, Leak_in_kitchen

Student ID:

(c) [10 marks] Suppose we include these five rules in a simplified backward chaining interpreter as follows:

```
rule1:: if kitchen_dry and hall_wet then leak_in_bathroom.
rule2:: if hall_wet and bathroom_dry then problem_in_kitchen.
rule3:: if problem_in_kitchen and no_water_from_outside then
        leak_in_kitchen.
rule4:: if window_closed then no_water_from_outside.
rule5:: if no_rain then no_water_from_outside.

confirm(Goal) :- Rule :: if Preconds then Goal, confirm(Preconds).
confirm(Goal1 and Goal2) :- confirm(Goal1), confirm(Goal2).
confirm(Goal) :- fact(Goal).

fact(hall_wet).
fact(bathroom_dry).
fact(no_rain).
```

Complete the code by defining the operators so that Prolog can understand these five rules.

Hint: operator definitions are of the form: `:- op(600, fx, in)`. 600 is higher priority than 610.

```
|:- op(611, xfx, ::).
|:- op(605, fx, if).
|:- op(603, xfx, then).
|:- op(601, xfy, and).
```

(d) [8 marks] Prolog scans the rules top down so that the priority of the rules can be changed by changing the order of the rules. Will the conclusion change for this problem if we are using a backward chaining interpreter and the rules above? Justify your answer.

|

Student ID:

Question 4. Planning

[35 marks]

(a) [4 marks] Name the four parts of the following STRIPS operator for cooking eggs in a frying pan. (Note: as in Prolog, variables are capitalised, and constants are lowercase):

- (A) fry-eggs(Eggs, Pan)
- (B) raw(Eggs), in(Eggs, Pan), on(Pan, stove)
- (C) raw(Eggs)
- (D) cooked(Eggs), hot(Pan)

(A) Name of action and parameters:
(B) Preconditions
(C) Deletes: facts deleted by the operation
(D) Adds: facts added by the operation

(b) [4 marks] The following (invalid) STRIPS operator describes the action of delivering a crate by truck from a depot to the crate's destination. The action includes the loading of the crate, but not the unloading (since crates on a truck may be unloaded separately). However, it has two errors. Write a corrected version of the operator.

```
transport-load(Crate, Depot, Address)
pre: at(Truck, Depot), at(Crate, Depot), delivery-destination(Crate, Address)
del: at(Truck, Depot)
add: in(Crate, Truck), at(Truck, Address)
```

The Truck variable is not mentioned in the parameters
The at(Crate,Depot) should be deleted

```
transport-load(Truck, Crate, Depot, Address)
pre: at(Truck, Depot), at(Crate, Depot), delivery-destination(Crate, Address)
del: at(Truck, Depot),at(Crate, Depot)
add: in(Crate, Truck), at(Truck, Address)
```

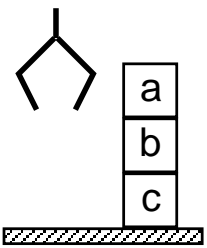
Student ID:

(c) [10 marks] Suppose a goal stack planner was given a blocks-world goal $\text{on}(b,a) \ \& \ \text{on}(a,c)$, and an initial state of $\text{clear}(a) \ \& \ \text{on}(a,b) \ \& \ \text{on}(b,c) \ \& \ \text{on}(c, \text{table}) \ \& \ \text{ae}$. At some point, it will reach the state shown on the facing page. Show the next steps of the planner up to the point that it has added three actions to the current plan. Make the order of the steps clear by numbering each change to the goalstack, the current state, or the current plan.

The four blocks-world operators that we used in lectures are as follows: (“ae” means “arm-empty”).

<pre>operator(stack(X,Y), pre: [clear(Y), holding(X)], del: [clear(Y), holding(X)], add: [ae, on(X,Y)]).</pre>	<pre>operator(putdown(X), pre: [holding(X)], del: [holding(X)], add: [on(X,table), ae]).</pre>
<pre>operator(unstack(X,Y), pre: [on(X,Y), clear(X), ae], del: [on(X,Y), ae], add: [holding(X), clear(Y)]).</pre>	<pre>operator(pickup(X), pre: [clear(X), on(X,table), ae], del: [on(X,table), ae], add: [holding(X)]).</pre>

Current Plan	Goal Stack
-2 unstack(a,b)	on(b,a) & on(a, c)
-7 putdown(a)	on(b,a)
10 unstack(b,c)	stack(b,a) clear(a) & holding(b) holding(b)
	-10 unstack(b,c)
	-9 on(b, c) & ae & clear(b)
	-3 clear(b)
	-2 unstack(a,b)
	-1 on(a, b) & ae & clear(a)
	-8 4 ae
	-7 5 putdown(a)
	-6 5 holding(a)



10 holding(b)
~~-7 2 holding(a)~~ ~~2 clear(b)~~ ~~7 on(a, table)~~ ~~7 ae~~
~~-2~~ ~~-10~~ ~~-2~~
Current state: clear(a) & on(a,b) & on(b, c) & on(c, table) & ae

Student ID:

(d) [5 marks] Given a planning graph with two actions A and B in the same level, there could be three different reasons why A and B are mutually exclusive (mutex). List the three reasons:

- | A and B have mutually exclusive effects
- | There are no mutually consistent sets of preconditions of A and B
- | An effect of A (B) negates a precondition of A (B)

Consider the following operators, written in graphplan format, for cutting grass with a mower, and putting the tools away in the shed, along with declarations, initial state, and goal.

```
(operator
  Cut-grass
  (params
    (<tool> Tool))
  (preconds
    (in <tool> garden)
    (tall grass))
  (effects
    (del tall grass)
    (short grass)))
```

```
(operator
  Putaway
  (params
    (<tool> Tool))
  (preconds
    (in <tool> garden))
  (effects
    (del in <tool> garden)
    (in <tool> shed)))
```

```
(operator
  Lock-Shed
  (params)
  (preconds
    (in mower shed))
  (effects
    (locked shed)))
```

```
Declarations:
  (mower Tool)
```

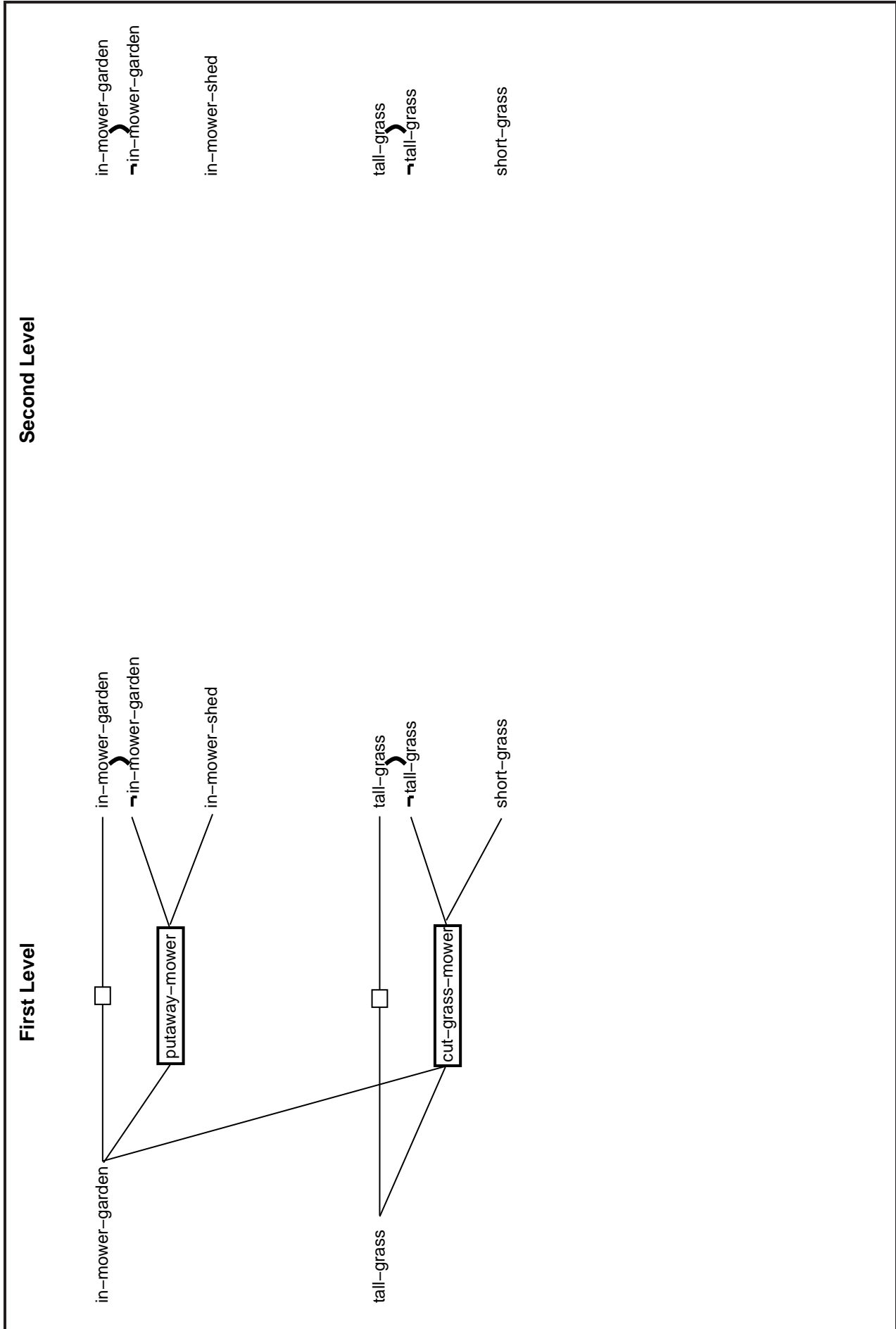
```
initial state:
  (preconds
    (in mower garden)
    (tall grass))
```

```
Goal:
  (effects
    (in mower shed)
    (short grass))
```

The diagram on the facing page shows part of the planning graph that graphplan would construct.

(e) [12 marks] Extend the planning graph:

- (i) Identify all pairs of mutex actions in the first level
- (ii) Identify all pairs of mutex facts in the first level
- (iii) Constructing all actions and their effects in the second level
- (iv) Identify all pairs of mutex actions in the second level



Question 5. Natural Language Processing

[23 marks]

Consider the following DCG grammar and lexicon for simple commands in an artificial game language with a lexicon similar to English, but with a different grammar:

- verbs come after the objects they should act on,
- numbers and adjectives come after the noun.

For example, the equivalent of the English “give three red helmets to the soldier” would be the command “helmet three reds soldier give.”

Grammar:

```

command --> command1, [then], command.
command --> command1.

command1 --> np, verb
command1 --> np, np, verb.

np --> noun, ndesc
np --> noun.

ndesc --> count.
ndesc --> adj.
ndesc --> count, adj.

verb --> [V], {lexicon(V, verb, -) }.
noun --> [N], {lexicon(N, noun)}.
count --> [C], {lexicon(C, count,-)}.
adj --> [A], {lexicon(A, adj,-)}.
    
```

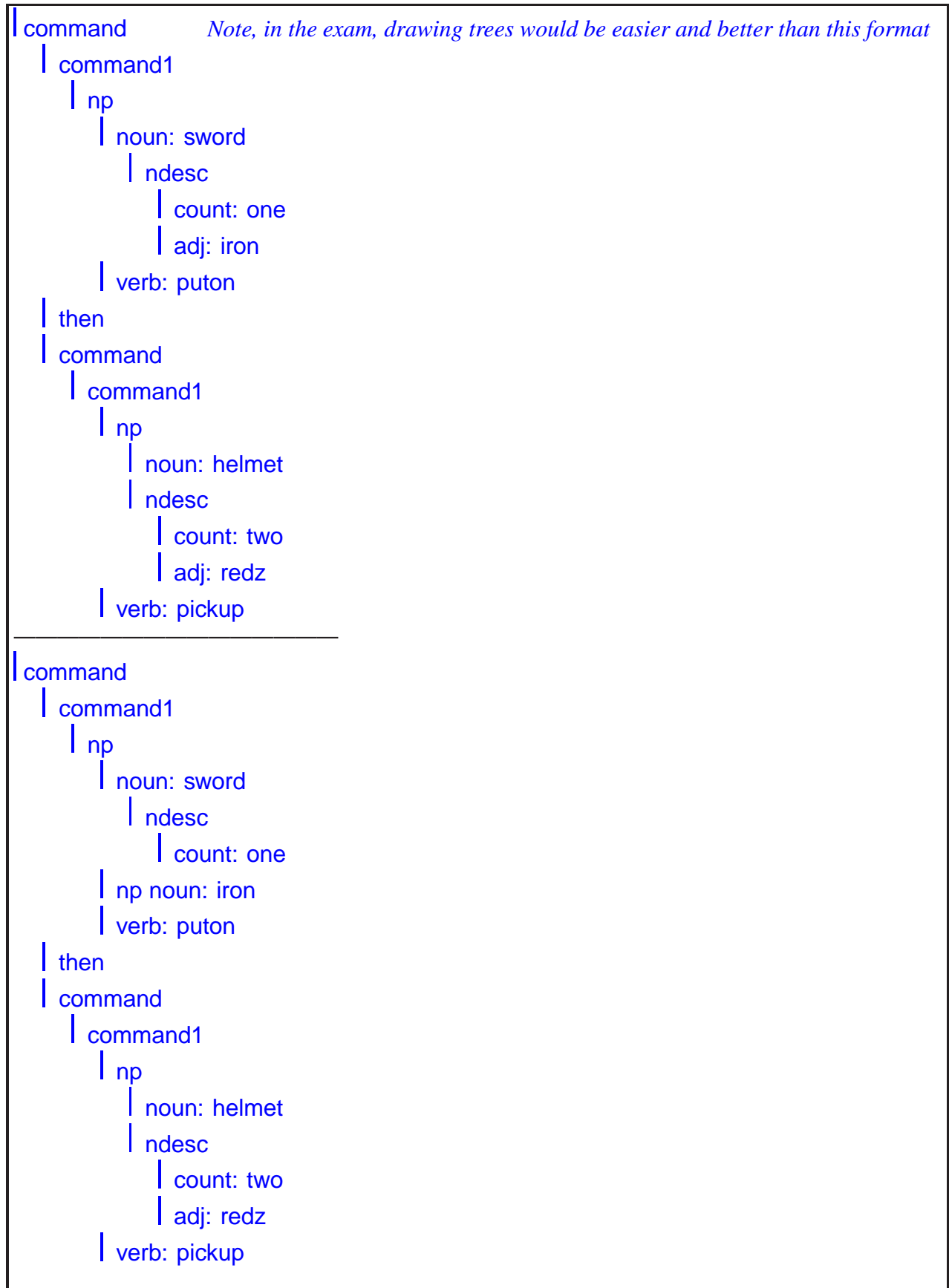
Lexicon:

```

lexicon(sword, noun).
lexicon(iron, noun).
lexicon(table, noun).
lexicon(helmet, noun).
lexicon(soldier, noun).
lexicon(box, noun).
    :
lexicon(one, count, singular).
lexicon(two, count, double).
lexicon(three, count, multi).
lexicon(four, count, multi).
    :
lexicon(pickup, verb, [one]).
lexicon(give, verb, [two]).
lexicon(puton, verb, [one, two]).
    :
lexicon(iron, adj, singular).
lexicon(ironz, adj, double).
lexicon(irons, adj, multi).
lexicon(large, adj, singular).
lexicon(largez, adj, double).
lexicon(larges, adj, multi).
lexicon(red, adj, singular).
lexicon(redz, adj, double).
lexicon(reds, adj, multi).
    :
    
```

Student ID:

- (a) [6 marks] Show all the parse trees for the following command:
“sword one iron puton then helmet three reds pickup.”



Student ID:

(b) [10 marks] The grammar above does not enforce two constraints of the language:

- Commands with some verbs (*eg pickup*) can only have one noun phrase, but commands with other verbs (*eg give*) must have two noun phrases. Other verbs (*eg puton*) can have either one or two noun phrases.
- Adjectives can be “singular” (*eg red*), “double” (*eg redz*), or “multi” (*eg reds*). An adjective in a noun phrase must agree with the count: if the count is **one** or there is no count, then the adjective must be “singular”, if the count is **two**, then the adjective must be “double”, and if the count is **three** or more, then the adjective must be “multi”.

For example, the sentences below with an * are invalid

helmet soldier give	* helmet box pickup	
* helmet give	helmet pickup	
belt one red puton	* belt two red puton	* belt three red puton
* belt two red puton	belt two redz puton	* belt three redz puton
* belt three red puton	* belt two reds puton	belt three reds puton

Modify the grammar to enforce these two constraints. Note that the lexicon already contains the information needed for these constraints.

You only need to show the changed rules.

```
| command1 --> np, verb(one).
| command1 --> np, np, verb(two).
| verb(Mode) --> [V], {lexicon(V, verb, Modes), member(Mode, Modes)}.
| ndesc --> count(_).
| ndesc --> adj(singular).
| ndesc --> count(N), adj(N).
| count(N) --> [C], {lexicon(C, count, N)}.
| adj(N) --> [A], {lexicon(A, adj, N)}.
```

Student ID:

The following newspaper headline is ambiguous and has at least three interpretations (not all of which are physically possible).

“Miners refuse to work after death”

(c) [2 marks] Explain two possible interpretations.

<ul style="list-style-type: none"> The miners are refusing to work after they have died One of the miners' colleagues has died and the miners are now refusing to work. The refuse produced by the miners is going to work after it has (or they have) died
--

(d) [5 marks] People can usually work out the intended meaning of this headline. Outline some of the knowledge an agent would need in order to work out which interpretation was intended by the writer.

<p> </p>

Question 6. Machine Learning

[36 marks]

(a) [4 marks] The Naïve Bayes algorithm uses Bayes’ rule

$$p(C|D) = \frac{p(D|C)p(C)}{p(D)}$$

to compute the probability of a class C given the data D . If the data is described by a vector of n attributes (A_1, A_2, \dots, A_n) , state how Naïve Bayes approximates the value of $p(D|C)$, and state what assumption it is making about the data.

$p(D|C) \approx p(A_1|H)p(A_2|H) \dots p(A_n|H)$

Assumption:
 It assumes that the attributes are conditionally independent of each other, given the class

(b) [10 marks] Suppose you are building a Naïve Bayes spam filter to distinguish spam messages from real email messages. You have picked two key words: “medicine”, and “assignment” to characterise each message, and have counted how many of the messages contain each word:

	spam		email	
	word present	word not present	word present	word not present
“medicine”	50	350	1	49
“assignment”	1	399	20	30
Total count	400		50	

If your spam filter was presented with a new message that contained the word “assignment” but did not contain “medicine”, would your spam filter classify the message as spam or as email? Show your working.

score(spam) = $350/400 * 1/400 * 400/450 = 350/(400*450) = 7/3600$
 score(email) = $49/50 * 20/50 * 50/450 = 98/(5*450)$ approx $2/45$
 Therefore, it will choose email

Student ID:

(c) [4 marks] Explain the requirements on a purity function, as used in the decision tree learning algorithm and how the $p(C_1)p(C_2)$ formula meets these requirements. State a limitation of this formula.

Explanation: | The purity function must be maximum when all the instances in a set
| are the same class and must be a minimum when there is an equal mix of instances
| from different classes (and should be smooth between)
| $p(C_1)p(C_2)$ is actually an impurity function.
| It is 0 when there is only one class, and a maximum when there is an equal mix
Limitation: | It only works with two classes

(d) [10 marks] Consider the following data set describing 10 menu items from a restaurant, of which 5 are popular with customers, and 5 are not. They are described by three attributes: whether they are spicy or mild, whether they have sauce or not, and whether the protein base is vegetarian, beef, or chicken.

Spice	Sauce	Protein	Category
spicy	yes	beef	Popular
spicy	yes	beef	Popular
spicy	yes	vegetarian	Popular
spicy	no	beef	Popular
spicy	no	beef	Popular
spicy	yes	beef	unpopular
spicy	yes	beef	unpopular
spicy	no	beef	unpopular
mild	no	beef	unpopular
mild	no	chicken	unpopular

Which attribute would the decision tree building algorithm choose for the root of a decision tree for predicting whether a menu item will be popular or unpopular? Show your working.

| spice: $8/10 * (5/8 * 3/8) + 2/10 * (0/2 * 2/2) = 3/16 = 6/32$
| sauce: $5/10 * (3/5 * 2/5) + 5/10 * (2/5 * 3/5) = 6/25$
| protein: $0 + 0 + 8/10 * (4/8 + 4/8) = 1/5 = 6/30$
|
| spice has the lowest score, therefore the algorithm will use
| spice at the root

Student ID:

(e) [3 marks] Perceptrons are a classification mechanism that use a set of weights and a threshold. The perceptron learning algorithm is able to learn a threshold and set of weights from a set of positive and negative examples. State the major limitation of the the perceptron learning algorithm.

The instances must be linearly separable (by a hyperplane)

(f) [5 marks] Explain how the hidden nodes of a multilayer feedforward neural network mean that it does not suffer from the same limitation as the perceptron.
