Final Exam

Aiden Allen

---

1 A) What is Artificial Intelligence?

Artificial Intelligence is the effort to simulate or supplement human intelligence using machines, particularly computers. It encompasses a wide range of technologies, such as large language models, expert systems, computer vision, and more. AI is particularly valuable for solving problems that exceed human capacity due to their complexity or scale, leveraging the speed and efficiency of computational systems.

At its core, AI can be described as a set of instructions or algorithms designed with built-in knowledge or goals, which guide its processing and decision-making to perform specific tasks.

1 B) Do you agree with Lovelace's statement in the context of modern day computing? Explain your answer, i.e., state why you agree or disagree.

Lovelace's statement regarding the analytical engine holds true today. Specifically in the context of the limitations of expert systems we can confirm her statement to hold true, *"it can do whatever we know how to order it to perform. It can follow analysis; but it has no power of anticipating any analytical relations or truths."*. Expert systems possess only the knowledge that is given to them, therefore they cannot draw any conclusions outside of this. For example if we do not provide the system with the information regarding blood pressure, an expert system will have no knowledge of the existence of this measurement nor the significance it may provide to analysis of some problems. AI systems maybe able to draw similarities and understand basic correlations between set features, however, it will not be able to provide any new information that could not be calculated manually. AI serves more to assist us in those calculations, than to draw new conclusions.

1 C) What is the Turing Test? What are some pitfalls of the Turing Test?

The Turing test, a test created by the great cryptographer Alan Turing in 1949, is an attempt to classify if machines are intelligent. The test has 3 components: the machine, a human subject, as well as a tester. The machine will attempt to simulate human intelligence with the tester, in an attempt to fool the testing into believing that it is the human subject. A machine is thus considered to be intelligent if it can fool the tester.

This test comes with many pitfalls, and has come under major scrutiny since its conception. For one, the goal of an intelligent machine is not necessarily to simulate human intelligence, rather to supplement it. Machines may possess their own independent form of intelligence, that may present differently than human intelligence. Additionally, humans are extremely variable making this test subjective and completely unscientific. One tester may reach a different conclusion than another. This is especially damning given the limited sample size of the test. These flaws not only challenge the validity of the test as a measure of intelligence but also underscore the need for more robust and scientific approaches.

1 D) What is the Physical Symbol Hypothesis, and Do you Agree with it?

The Physical Symbol Hypothesis, proposed by Alan Newell and Herbert Simon, argues that for a machine to be considered intelligent, it must be able to understand and use physical symbols. I don't agree with this conclusion for reasons similar to the issues with the Turing Test. The hypothesis overlooks how difficult it is to define

what "understanding" really means. For example, the Chinese Room argument shows that someone could manipulate Chinese symbols to produce valid responses without actually understanding the language, as long as they have a set of rules to follow. Similarly, the Physical Symbol Hypothesis might oversimplify what intelligence actually is.

1 E) In 1980, John Searle presented his Chinese Room Argument. What is the key point of Searle's argument? Do you agree with it?
The Chinese room argument posits that given a set of rules, a person maybe able to produce and manipulate Chinese characters in a manner in which it seems as though the subject understands what they are doing, even though they are just following a set of rules with no real understanding. This test is an argument against the possibility of machine understanding, and thus a counter to the physical symbol hypothesis. I do agree with this argument, given that in many ways it seems machines do not really understand what are are doing. For instance code generation, the major pitfall of LLM's are their inability to practically understand what they are doing. This can be demonstrated in the form of hallucinations as well as generating code that misuses libraries.

---

2 A) The Resolution inference rule says “From P ∨ Q and ¬Q ∨ R infer P ∨ R. Give a truth table argument that verifies that this rule is sound.

P	Q	R	P ∨ Q	¬Q ∨ R	P ∨ R
T	T	T	T	F	T
T	T	F	T	F	F
T	F	T	F	T	T
T	F	F	F	F	F
F	T	T	F	F	F
F	T	F	F	F	F
F	F	T	F	T	F
F	F	F	F	F	F

2 B) Give a semantic argument that explains why ∃XP (X) is logically equivalent with ¬∀X¬P(X)
First we will lay each argument out verbally to better understand the proposed argument.

- $\exists X P(X)$: There Exists X such that P(X) is True, meaning that there is at least one element X such that P(X) is true
- $\neg \forall X \neg P(X)$: Not For All X Not P(X), meaning that it is not the case that for all X P(X) is false. In other words, there must be some X that makes P(X) true.

Therefore if within, $\exists X P(X)$, we are stating that there must be an X that leads P(X) to be true. This is logically equivalent to saying that it is not the case that all P(X) is false. Since P(X) being false in all cases would negate the former. Therefore we have proved these statements logical equivalence.

2 C) if X does not occur free in Q, then ∀X(P (X) → Q)
is semantically equivalent with ∃XP (X) → Q. Give an example to show that these formulas will not necessarily be equivalent if X does occur free in Q.
Lets first define our predicates to try and further understand why the formulas will not necessarily be equivalent if X does occur free in Q.

- P(X): X is a cat
- Q: X is small

Now in understanding our question further lets express it in terms of our predicate definition.

- for all X, if X is a cat, then X is small
  - Stating that if X is a cat it is small. In other words all cats are small, and if X is a cat than it must be small
- if there exists an X such that X is a cat then X is small.
  - Stating much the same, with the exception of the existence of X

Now we can identify the difference between these statements to be that the latter requires the existence of X, whereas Q is not bound to any specific X. It's a global statement that claims there's some small X if there exists a cat, which does not specify which cat must be small.

- P(a) -> a is a cat
- P(b) -> b is not a cat
- Q(a) -> a is small
- Q(b) -> b is not small

For ∀X(P (X) → Q):

- P(a) is true, thus Q(a) must be true.
- P(b) is false

For ∃XP(X)→Q:

- The premise ∃XP(X) is true, since there exists and X such that P(a) holds true.
- The conclusion Q, implies all X are small, However Q(b) is false. So implication fails

So they are not semantically equivalent, since the first formula holds true in all cases, and the latter does not.

---

3 A) Consider the expressions ancestor(X, Y) and ancestor(dan, aaron). Give a unifying substitution, if there is one, or explain why they can't be unified if there isn't.
Unification is the process of finding a substitution of variables that makes the terms the same. To unify the given statements we need to find a substitution for X and Y such that ancestor(X, Y) becomes ancestor(dan, aaron).

We can do this through `X = dan` and `Y = aaron`. Thus the unification is then `{X/dan, Y/aaron}`. In other words, we can substitute X for dan and Y for aaron, and conclude a unifyed solution does indeed exist.

3 B) Consider the expressions ancestor(X, father(X)) and ancestor(aaron, dan), and assume it is known that dan = father(aaron). Give a unifying substitution, if there is one, or explain why they can't be unified if there isn't.
We know the the X within `ancestor(X, father(X))` to be `aaron`, given that `dan = father(aaron)`. Since we know that `father(aaron) = dan`, we can substitute the statement `father(aaron)` with `dan`. This leaves us with `ancestor(aaron, dan)`. Thus the unifying statement is `{X/aaron}`.

3 C) Give the composition of the substitutions {a/Z, Y /X} and {Z/W, b/Y} in that order.

1. First Substitution

$$a/Z, Y/X$$

The given substitution means that we may substitute all occurrences of `Z` with `a`, and all occurrences of `X` with `Y`.

2. Second Substitution

$$Z/W, b/Y$$

   The given substitution means that we may substitute all occurrences of `W` with `Z`, and all occurrences of `Y` with `b`.

When applying these substitutions, the fist substitution contains `Z` so we will replace its occurrence with `a`, then `Z` is also associated with `W`. So we will replace `Z` with `W`, but since we have already replaced it with `a` we will replace it then apply our second substitution. This will apply no change at all since our `Z` was already replaces by `a`.

In replacing `b/Y`, `Y` is associated with `X`, as well as with `b`. We will replace it with `b` since we are using the second substitution. Thus applying our first substitution will yield:

$$a/Z, X/b, Y/X$$

And our second substitution will result in the following:

$$a/Z, X/b, Y/X$$

---

4 A) Write a statement that says that Jill is female.
We can say that Jill is female through the following Prolog statement: `female(Jill)`.

4 B) Write a statement that says that John is a parent of Jill.
We can say that John is the parent of Jill through the following Prolog statement: `Parent(John, Jill)`

4 C) Write a statement that says that X is the mother of Y if X is a parent of Y and X is female

Below is the statement expressing that if a parent is female than they are the mother.

$$\forall X \forall Y (parent(X, Y) \land female(X) \implies mother(X, Y))$$

4 D) Write a statement that says that X and Y are siblings if they have a common parent

The statement below outlines that if there exists a Z such that the parent of both X and Y is Z, and X and Y are not the same person, then this would imply that they are siblings.

$$\forall X \forall Y (\exists Z (parent(Z, Y) \land parent(Z, Y) \land X \neq Y)) \implies sibling(X, Y)$$

4 E) Write a pair of statements that define the ancestor relation in terms of the parent relation.

Since this problem requires a recursive solution we need to begin by defining our base case. This will show that an ancestor is directly related to X, and Y.

$$\forall X \forall Y (parent(X, Y) \implies ancestor(X, Y))$$

In order to further our search space, adding non direct ancestors, we need to implement a recursive case.

$$\forall X \forall Y \forall Z (parent(X, Z) \land ancestor(Z, Y) \implies ancestor(X, Y))$$

4 F) Write a statement that says that everyone has an ancestor

$$\forall X \exists Y (ancestor(Y, X))$$

4 G) Write a statement that says that no one can be his or her own ancestor.

$$\forall X (\neg ancestor(X, X))$$

---

5 A) What is the size of the domain of the smallest possible model for the statements a-g.

First we need to define the constraints given in statements a-g. These are as follows

1. Jill must be female
2. One parent must exists of Jills
3. The ancestors are describer transitively
4. We cannot have someone be their own ancestor

Therefore we can see the minimum domain using the constraints above to be 2. Since we are only required to have a female Jill, and a parent of Jills.

5 B) What is the size of the domain of the smallest possible model for the statements a-f

Removing the *looping constraint* we are now allowed to state that X can be an ancestor of X. This means that X can be their own parent, because of this our space is reduced to 1. However if we are to require X to not equal Y then our space will remain 2.

---

6 A) Write Prolog statements that express as many as possible of the statements described in Question 4 (parts a through g), and, for any statement that cannot be so expressed, explain why.

```
% jill is a female
female(jill).
```

```
% john is a parent of jill
parent(john, jill).
```

```
% x and y are siblings if they have a common parent %
sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.
```

```
% base case
ancestor(X, Y) :- parent(X, Y).
```

```
% recursive case
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

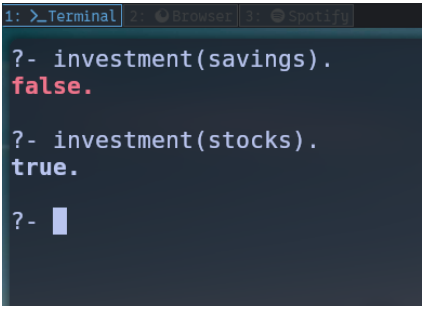
```
% everyone must have an ancestor
ancestor(_, X).
```

```
% you cannot be your own ancestor
:- ancestor(X, X).
```

---

7 A) The Prolog implementation of this system is given in Chapter 2 of the supplementary text. Turn in a copy of this program but where the individual in concern has 2 dependents, a steady income of $30,000 per year, and $20,000 in savings. Run this program and give a screen shot showing the result.

```
investment(savings) :- savings_account(inadequate).

investment(stocks) :- savings_account(adequate), income(adequate).

investment(combination) :- savings_account(adequate), income(inadequate).

savings_account(adequate) :- amount_saved(X), dependents(Y), minsavings(Y,
B), X > B.

savings_account(inadequate) :- amount_saved(X), dependents(Y), minsavings(Y,
B), not(X > B).

income(adequate) :- earnings(X, steady), dependents(Y), minincome(Y, B), X >
B.

income(inadequate) :- earnings(X, steady), dependents(Y), minincome(Y, B),
not(X > B).

income(inadequate) :- earnings(_, unsteady).

minsavings(A, B) :- B = 5000 * A.

minincome(A, B) :- B = 15000 + (4000 * A).

amount_saved(22000).

earnings(25000, steady).

dependents(3).
```



7 B) Write a production system using the previous part

1. If the savings account is inadequate then the investment advice is *savings*
2. If the savings account is adequate and the income is adequate then the investment advice is *stocks*
3. If the savings account is adequate and the income is inadequate then the investment advice is *combination*
4. If the amount saved is greater than the minimum savings then the savings is *adequate*
5. If the amount saved is less than or equal to the minimum savings then it is *inadequate*
6. If the income is steady and the income is greater than the minimum required income then it is *adequate*
7. If the income is steady and the income is less than or equal to the minimum required income then it is *inadequate*
8. If the income is not steady (unsteady) then it is *inadequate*
9. The minimum savings is the number of dependents times 5000 ($dependents * 5000$)
10. The minimum income is the number of dependents times 4000 plus 15000 ($(dependents * 4000) + 15000$)

7 C) State Space For the Previous Problem

There are 3 possible results; *Stocks*, *Combination*, and *Savings*. From these we can gather the possibilities that go into each of these combinations.

Situation 1:

- Savings: 20,000
- Earnings: 30,000
- Dependents: 2
- MinSavings: $10,000 = 5000 \times 2$
- MinIncome: $23,000 = (4000 * 2) + 15000$
- Investment: Stocks

Since as we can see the savings and earnings are both considered to be adequate this will result in the investment advice being stocks

Situation 2:

- Savings: 0
- Earnings: 30,000
- Dependents: 2
- MinSavings: $10,000 = 5000 \times 2$

- MinIncome: $23,000 = (4000 * 2) + 15000$
- Investment: Savings

Since the minimum savings has not been met we can conclude this situation to resolve in savings.

Situation 3:

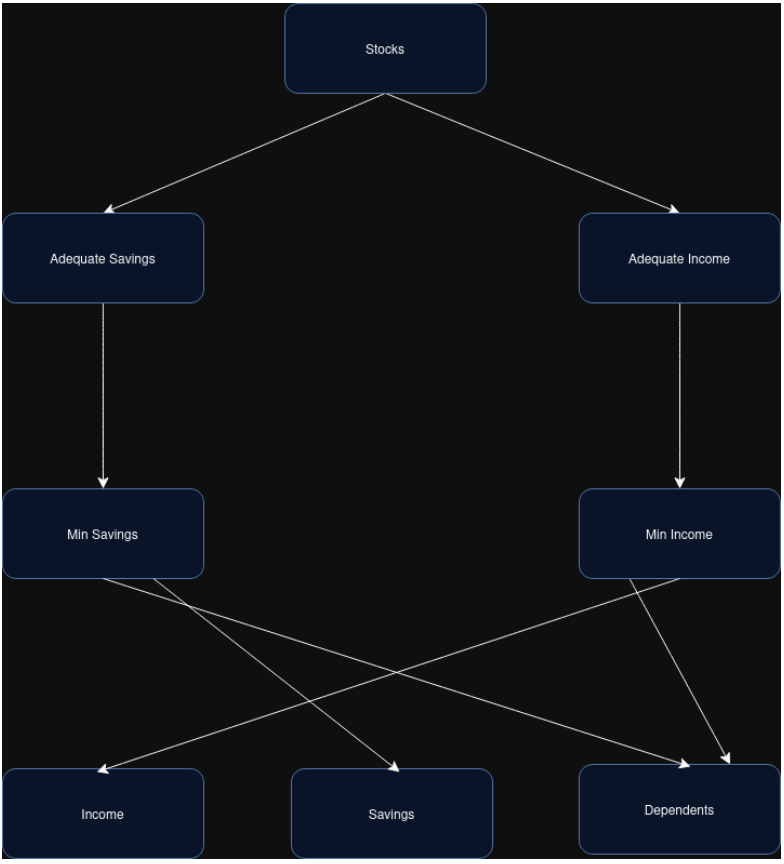
- Savings: 20,000
- Earnings: 10,000
- Dependents: 2
- MinSavings: $10,000 = 5000 \times 2$
- MinIncome: $23,000 = (4000 * 2) + 15000$
- Investment: Combination

Since the minimum savings has been met and the minimum income has not the investment advice will be combination.

We can now determine all of the possible combinations to be:

- Stocks
  - Adequate Savings and Adequate Income
- Savings
  - Inadequate Savings
- Combination
  - Adequate Savings and Inadequate Income

7 D) Diagram and Unifying Statements



For the example used our unifying substitutions will be

- Savings: {X = 20000, Y = 2}
- Income: {X = 30000, Y = 2}

7 E) Resolution Refutation

Using these facts:

- Amount saved: 20,000
- Earnings: 30,000 and Steady
- Dependents: 2

We can now calculate the minimum savings and income to be 10,000 and 23000 respectively, and expand these definitions to include this calculation.

1. $\neg investment(stocks) \vee savingsaccount(adequate)$
2. $\neg investment(stocks) \vee income(adequate)$
3. $\neg savingsaccount(adequate) \vee (amountsaved(X) \wedge dependents(Y) \wedge X > minsavings(Y,))$
4. $\neg income(adequate) \vee earnings(X, steady) \wedge dependents(Y) \vee X > minincome(Y,)$
5. $\neg investment(X)$

We now need to assume that the investment advice is $\neg investment(stocks)$. Then we must apply $\neg investment(stocks) \vee savingsaccount(adequate)$. We can see from applying the third rule as well that $\neg(amountsaved(20,000) \wedge dependents(2) \wedge 20000)$ is greater than the target $10,000$. Therefore the savings is adequate. We can not see that $\neg investment(stock)$ is viable.

---

8 A) What is a heuristic?

A heuristic is a guiding rule, essentially used to guide the program in decision making. This makes for more efficient traversal since there are rules in place to guide decisions, ignoring or skipping past non relevant paths within search.

8 B) What is represented by the two main components of a heuristic evaluation function.

The cost so far $g(n)$. This represents the actual cost from the start node to the node n. For instance this can represent the number of nodes needed for travel from the start node to the current node.

The cost to the goal $h(n)$. This represents the cost from the node n to the goal node. This is an estimate, and it will inform our search on moving efficiently.

Combined these form $f(n) = g(n) + h(n)$.

8 C) What is an algorithm of type A?

An algorithm of type A commonly refers to an algorithm which is guided by an evaluation function within a state space search. This algorithm uses a best first search in order to prioritize exploration of some nodes, in order to more efficiently reach the solution node.

8 D) What is an algorithm of type A*?

An algorithm of type A* commonly refers to informed algorithms that attempt to find the optimal solution path within a search space. This is accomplished through the combination of cost search and a greedy best first search, utilizing a heuristic evaluation to guide the search path.

8 E) How is the “open” list managed in depth-first, breadth-first, and best-first searches?

Within a Depth First Search (DFS), a *stack* is used to manage the open list. Within this stack new nodes are added to the top of the list and removed from the top as well. This is more memory efficient as only the currently traversed path needs to be stored, however it maybe less efficient in finding the goal path since only one path may be searched at a time.

Within a Breadth First Search (BFS), a *queue* is used to manage the open list. Within a queue the current state is added to the bottom of the list, while states are removed from the top. A queue will allow us to traverse all states at the same level, allowing us to search each path in parallel until the goal path is reached. This comes with the advantage that the shortest path (unweighted) to the goal node will be found, but also the disadvantage of utilizing more memory than the former.

Within a Best First Search (BFS), a *priority queue* is used to manage the open list. A priority queue will remove only the highest (or lowest given a min heap) heuristic evaluation state. This will allow for prioritizing the most promising paths, gaining an advantage over DFS efficency and BFS memory usage. A best first search requires a heuristic evaluation function in order to determine the promise of each node, in guiding the search space.

---

9 A) Which of these four heuristics are monotone? Explain why you think this (if possible, give a proof). If you don't have a complete or precise explanation, give your best opinion.

In the example of *tiles out of place*, we would consider this to be monotone since moving a tile closer to the goal position will decrease the cost to the goal by at most 1, which is the same if not less than the cost of moving one tile.

The *Sum of the distance out of place* heuristic is also monotone. This is because the distances will always be decreased by one, the same as the cost needed to reach the goal position.

The *2 × the number of direct tile reversals* is however not monotone. This is because moving one tile may result in tiles needing multiple reversals, increasing the estimated cost given by the heuristic.

Since a *constant 0* is the same for all states this heuristic would be considered monotone as well.

9 B) Which are admissible? Again, explain why you think this?

A heuristic is considered to be admissible if it never overestimates the cost. Using this knowledge we can now classify each heuristic on whether it is admissible or not.

Tiles out of place is considered to be admissible since it will only count misplaced tiles. In this way it will always underestimate the cost, or exactly estimate the cost. This allows us to determine that it is in fact admissible.

Sum of the distance out of place is also admissible. It will provide us with the lower bound of moves needed, since it will only cont the distances out of place. This means that it will also only exactly or underestimate the cost needed.

2 x The number of tile reversals is probably admissible, however in some cases it will not be. In cases such as when only one tile may need to be reversed it may overestimate the moves needed.

Constant 0 is also admissible. This is because it will always provide the same result, which is a severe underestimation, and since 0 will always be less than the actual cost.

9 C) Rank the three heuristics in terms of their informedness (least to most)?

Constant 0 is the least informed heuristic since it has no knowledge of the state space at all. The result will always be constant meaning that it is not informed.

2 x the number of tile reversals is the second least informed, since reversals may play a key part in solving the puzzle however it is not necessarily the goal. This may leave the heuristic over evaluating and undervaluing particular states which may serve to better guide the decision.

Tiles out of place is the second most informed heuristic, this is because it does have knowledge of the space. However, its knowledge maybe counter intuitive in cases that require more tiles to be out of place in order for a more cost efficient path to be taken.

Sum of distance out of place is the most informed heuristic. This is because it has knowledge of both the goal state and the current state, as well as the relative distance from that state. In this way the heuristic is fairly informed.

9 D) Is a monotone heuristic necessarily admissible? Explain your answer.

Yes! Since a monotone heuristic is bound by the actual cost and the cost estimate it must be admissible. Since it cannot exceed the cumulative cost it is therefore assuredly admissible.

9 E) Is an admissible heuristic necessarily monotone? Explain your answer.

No! An admissible heuristic is not necessarily monotone, since individual moves maybe overestimated, however the total cost may not be.

9 F) Why is the constant 0 heuristic equivalent to breadth-first search?

First lets describe both a BFS and a best first search to better understand the problem statement.

A *breath first search* explores every possible state in parallel until a goal state is reached. This means that the shortest path will always be found.

A *best first search* is operates similarly to to a breadth first search while using a heuristic to guide the search process. This allows us to remove paths that are deemed unnecessary, thus making for a more efficient search.

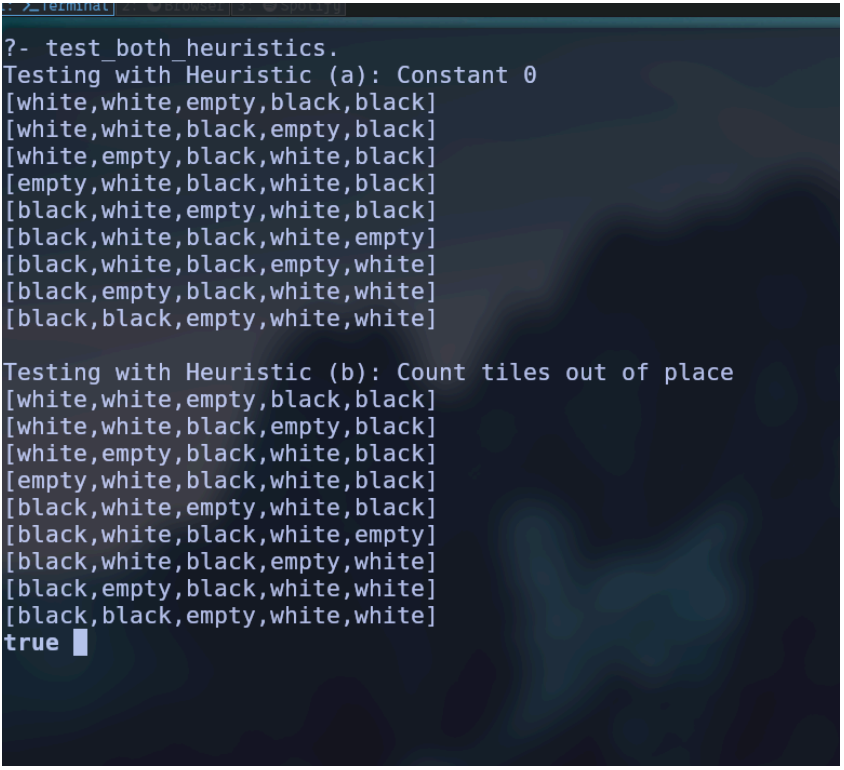
So as we can see through these definitions a best first search without a heuristic to guide its search will operate in the same way as a breadth first search. Since a constant heuristic will always result in the same evaluation it will not eliminate any pathways.

---

10 A) Sliding Puzzle Heuristic

```
goal([white, white, empty, black, black]).

heuristic_a(_, 0).
heuristic_b(State, H) :-
    goal(Goal),
    count_out_of_place(State, Goal, H).

count_out_of_place([], [], 0).
count_out_of_place([X|Xs], [Y|Ys], H) :-
    (X \= Y, X \= empty -> count_out_of_place(Xs, Ys, H1), H is H1 + 1;
    count_out_of_place(Xs, Ys, H)).

move([empty, X | Rest], [X, empty | Rest]).
move([X, empty | Rest], [empty, X | Rest]).
move([empty, X, Y | Rest], [Y, X, empty | Rest]).
move([X, Y, empty | Rest], [empty, Y, X | Rest]).
move([H | T1], [H | T2]) :- move(T1, T2).

best_first(Start, Heuristic, Path) :-
    search([[Start]], Heuristic, Path).

search([[Node | Path] | _], _, [Node | Path]) :-
    goal(Node).
search([[Node | Path] | Rest], Heuristic, Solution) :-
    findall([Next, Node | Path],
```

```
            (move(Node, Next), \+ member(Next, [Node | Path])),
            Children),
    evaluate_and_sort(Children, Heuristic, NewChildren),
    append(Rest, NewChildren, NewAgenda),
    search(NewAgenda, Heuristic, Solution).

evaluate_and_sort(Nodes, Heuristic, SortedNodes) :-
    maplist(evaluate(Heuristic), Nodes, EvaluatedNodes),
    sort(1, @=<, EvaluatedNodes, SortedEvaluatedNodes),
    pairs_values(SortedEvaluatedNodes, SortedNodes).

evaluate(Heuristic, [Node | Path], H-[Node | Path]) :-
    call(Heuristic, Node, H).

print_solution([]).
print_solution([State | Rest]) :-
    write(State), nl,
    print_solution(Rest).

test_constant_heuristic :-
    write('Testing with Heuristic (a): Constant 0'), nl,
    best_first([black, black, empty, white, white], heuristic_a, Path),
    print_solution(Path).

test_heuristic_b :-
    write('Testing with Heuristic (b): Count tiles out of place'), nl,
    best_first([black, black, empty, white, white], heuristic_b, Path),
    print_solution(Path).

test_both_heuristics :-
    test_constant_heuristic,
    nl,
    test_heuristic_b.
```



---

11 & 12) Undergraduate Student in CAP4601

---

13 A) Why is the language of first-order predicate calculus the basis for most knowledge representation schemes

First and foremost the language is expressive, it is fairly simple to understand. There are ways of expressing entities, such as `Jill` and `John`, as well as relations between these entities, `parent(Jill, John)`. Additionally there are ways of representing qualifiers, such as exists (∃) and for all (∀), and properties of these, (`female(Jill)`).

Secondly, there are well defined semantics that provide a clear manner for interpretation as well as it is domain independent. It relys on logical knowledge and reasoning, which can be used within any medium.

13 B) What was Alan Newell and Herbert Simon's Logic Theorist?

Alan Newell and Herbert Simon's *logic theorist* was one of the very first artificial intelligence programs. It was developed in 1956, and serves as a foundational achievement within the field of Artificial intelligence.

The project was developed with the purpose of simulating human intelligence by solving mathematical theorems. The program used symbolic logic as well as search

techniques in order to simulate human intelligence.

13 C) What is a semantic network?

A semantic network is a representational model that is used to express relationships between entities. Semantic networks are often used to represent complex models such as Artificial intelligence, creating a simpler view of the problem.

13 D) What was the aim of Roger Schank's conceptual dependency theory?

Roger Shank's *conceptual dependency theory* was developed in the 1970s as an attempt to provide a framework in order to model natural languages, in a manner that was language independent. The project aimed to enable computers to better understand human language in an intelligent manner.

13 E) What was the purpose of Roger Schank's theory of scripts?

The purpose of Robert Schrank's *theory of scripts* was to model how humans organize and predict events based on recurring patterns and knowledge. It was designed in order to help machines understand human phrases, and knowledge.

13 F) What are Marvin Minsky's frames?

Marvin Minsky frames are structures that attempt to model how humans understand information about the world. They are useful in teaching artificial intelligence reasoning, memory, as well as natural language.

13 G) What is John Sowa's theory of Conceptual Graphs?

John Sowa's theory of *Conceptual Graphs* is a method of expressing formal reasoning within a graph like structure. It allows us to more effectively express relationships between entities.

13 F) What was the aim of Doug Lenat's CYC project?

The aim of Doug Lenats CYC project was to create a knowledge base that will allow for understanding and reasoning to be implemented within artificial intelligence systems. It aimed to solve issues of AI not being able to understand apriori knowledge.

13 G) What is IBM's Watson known for?

It was known for its natural language processing, and data analysis capabilities. Watson was able to understand images, audio, as well as text. It operated in a manner that imitated human computation as well, even winning the show Jeopardy!

---

14 A) Why are expert systems referred to as a variety of strong problem solving?

Expert systems are referred to as a form of strong problem-solving because they are trained on domain specific tasks. Expert systems posses a large amount of knowledge about a specific task and know nothing outside of them, this additionally limits the search space making it more effective at solving domain specific problems. They typically posses logical problem solving skills within these specific domains, making them extremely effective problem solving within their specified domain.

14 B) In the construction of expert systems, what is the advantage of separating the inference engine from the knowledge base?

There are a few advantages to seperating the inference engine from the knowledge base within an expert system.

For one, it allows for the expert system to be *reusable* across multiple problems. For example the same expert system maybe effective at solving financial problems as well as medical ones depending on their knowledge base.

Additionally, This allows for expert systems to become more *scalable*. Rather than restricting the inference engine to a static knowledge base it can become more dynamic when separated, without requiring the system to be redesigned.

Modularity is another advantage to separating the knowledge base from the inference engine. This allows the system to be tested on a variety of knowledge bases, potentially leading to higher rates of success within the problem space.

---

15 A) Explain what is meant by nonmonotonic reasoning

Non-monotonic reasoning is a type of reasoning that allows for changes within the conclusion based on newly acquired knowledge. For example, if we are to state all birds can fly, then we revise our statement due to finding out penguins cannot fly. Since we changed our conclusion based on newly acquired knowledge this is considered to be non monotonic reasoning.

15 B) What is the role of nonmonotonic reasoning in planning?

Non-monotonic reasoning allows us to handle problems such as reasoning and uncertainty within computational systems. Since we can adjust our conclusions based on newly acquired knowledge this allows us to later redact our statements and learn from them. Non-monotonic reasoning allows our systems to be more dynamic and adapt to newly presented information in a more effective manner.