NAME: ....

STUDENT NUMBER: ....

Question 1, State Space and Heuristics, (10 marks)

In the context of the 8-puzzle problem give a description and a data structure for the following:

i) State space representation.

ii) Legal move representation.

Solution

i) We could represent any state of the 8-puzzle by a permutation of the nine integers, $[0, 1, 2, 3, 4, 5, 6, 7, 8]$. The position of the 0 integer in the permutation indicates the open slot in the puzzle.

For example: The start and goal states

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

could be represented by the permutations

$[2, 8, 3, 1, 6, 4, 7, 0, 5]$ and $[1, 2, 3, 8, 0, 4, 7, 6, 5]$

Full *state space* is then the tree of reachable states starting from the initial state and with each state connected by a legal move. It is not necessary to store the full state space in order to solve the 8-puzzle, it is only necessary to search state space for the goal state.

ii) A move for the 8-puzzle is represented by moving the empty slot in one of 4 directions, `[U,D,L,R]`, The move is legal if the direction selected does not move the empty slot off the 3x3 grid.

Question 2, Search Algorithms, (10 marks)

Using *Best First Search*, construct an A^* algorithm for the *8-puzzle* problem and explain why your algorithm is indeed A^*.

Solution

```
def bestfs(start,goal)
    open = [start],
    close = [],
    State = failure;
    while (open <> []) AND (State <> success) begin
        remove the leftmost state from open, call it X;
        if X is the goal, then
            State = success
        else begin
            generate children of X;
            for each child of X
              do case
                the child is not on open or closed
                    assign the child a heuristic value,
                    add the child to open,
                the child is already on open
                    if the child was reached by a shorter path then
                        give the state on open the shorter path
                the child is already on closed:
                    if the child is reached by a shorter path then
                        remove the state from closed and add the child to open;
              endcase
            endfor
            put X on closed;
        end;
        re-order states on open by (path_length_so_far + heuristic) merit (best leftmost);
    endwhile;
    return State;
end.

def heuristic(state,goal)
    return(number_tiles_out_of_place(state,goal))
end.
```

This algorithm is A^* because the next state searched has minimum path-length-so-far + heuristic from here-to-goal and the heuristic under-estimates the actual number of moves required to take the current state to the goal state.

Question 3, Game Playing, (10 marks)

Consider Grundy's two player version of **NIM** starting with **8-counters** in one pile.

a) Draw the full game tree.
b) Use the **minimax** search algorithm to show that **max** can always win if he plays first.

Solution

See notes......

Question 4, Prolog, (10 marks)

Consider the following first verse of a well-known hillbilly poem by *Moe Jaffe*:

> Many many years ago, when I was twenty-three,
> I married the widow, *Hoe*, as pretty as can be,
> The widow had a grown-up daughter, *Roe*, with hair of red,
> My father, *Poe*, fell for her and soon the two were wed.

a) Construct Prolog predicates to capture the relationships in the poem.

b) Construct a `grandfather` predicate in the loose sense that allows for step-relationships.

c) Explain how you would use your `grandfather` predicate to prove that *Moe* was now his own grandfather.

Solution

```
male(moe).
male(poe).
female(hoe).
female(roe).

spouse(moe,hoe).
spouse(hoe,moe).
spouse(poe,roe).
spouse(roe,poe).

parent(moe,poe).
parent(roe,hoe).
stepParent(X,Y) :-
        parent(X,W),
        spouse(W,Y).

grandfather(X,Y) :-
        male(Y),
        (parent(X,W);stepParent(X,W)),
        (parent(W,Y);stepParent(W,Y)).
```

Question 5, Logic, (10 marks)

Convert the following statements into WFFs.

```
1) Animals, that do not kick, are always unexcitable
2) Donkeys have no horns
3) A buffalo can always toss one over a gate
4) No animals that kick are easy to swallow
5) No hornless animal can toss one over a gate
6) All animals are excitable, except buffalo
```

and the use resolution to show that

```
Donkeys are not easy to swallow.
```

Solution

1)	$\forall x :\sim kicker(x) \rightarrow\sim excitable(x)$	$kicker(a_1)\vee \sim excitable(a_1)$
2)	$\forall x : donkey(x) \rightarrow\sim horned(x)$	$\sim donkey(a_2)\vee \sim horned(a_2)$
3)	$\forall x : buffalo(x) \rightarrow tosser(x)$	$\sim buffalo(a_3) \vee tosser(a_3)$
4)	$\forall x : kicker(x) \rightarrow\sim swallow(x)$	$\sim kicker(a_4)\vee \sim swallow(a_4)$
5)	$\forall x :\sim horned(x) \rightarrow\sim tosser(x)$	$horned(a_5)\vee \sim tosser(a_5)$
6)	$\forall x : excitable(x) \rightarrow\sim buffalo(x)$	$\sim excitable(a_6)\vee \sim buffalo(a_6)$
7)	$\forall x :\sim buffalo(x) \rightarrow excitable(x)$	$buffalo(a_7) \vee excitable(a_7)$
	negate the goal	
8)	$\sim (\forall x : donkey(x) \rightarrow\sim swallow(x))$	$donkey(a)$
9)		$swallow(a)$
	resolution	
10)	8 and 2	$\sim horned(a)$
11)	10 and 5	$\sim tosser(a)$
12)	11 and 3	$\sim buffalo(a)$
13)	12 and 7	$excitable(a)$
14)	9 and 4	$\sim kicker(a)$
15)	14 and 1	$\sim excitable(a)$
16)	15 and 13	$FALSE$

Attachments

Summary of first order logic algebra:

$$
\begin{aligned}
A \rightarrow B &\equiv (\sim A) \vee B \\
\sim (\sim A) &\equiv A \\
\sim (A \wedge B) &\equiv (\sim A) \vee (\sim B) \\
\sim (A \vee B) &\equiv (\sim A) \wedge (\sim B) \\
\sim (\forall x : P(x)) &\equiv \exists x : \quad \sim P(x) \\
\sim (\exists x : P(x)) &\equiv \forall x : \quad \sim P(x)
\end{aligned}
$$