1. Model the following domain in PDDL:

   A robotic agent needs to take care of a very big house. The house has 3 floors, and it is arranged as in the figure.

   The robot needs to do the following tasks:
   - clean the kitchen (floor 0);
   - clean the bedroom (floor 1);
   - clean the attic (floor 2).

   In addition to that, he has to wash the 2 stairs. To clean the rooms, he uses a vacuum cleaner (located in the room **floor1-0**). To wash the stairs, he uses a mop (located in the room **floor2-1**). The robot starts with no item and can carry only one item at the time. Once a stair is washed, the robot cannot use it anymore (otherwise it gets dirty again!). To clean the stairs, the robot must be in the room above them). Eventually, the two items must be deposited in the storage room to consider the task completed.
2. Write the PDDL problem file for the problem.
3. Give an example of a plan that solves the problem that you just modeled.

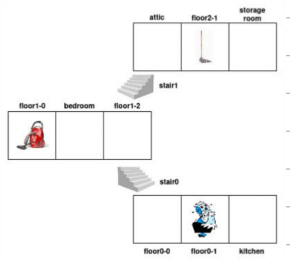


Figure: House domain.

```
(:requirment :adl)

(:types location
        stairs - location
        room - location
        vacuum
        mop
)

(:predicates (pos ?L - location)
             (dirtyS ?S - stairs)
             (dirtyR ?R - room)
             (at ?I - item ?L - location
             (busy)
             (adj ?L1 ?L2 - location)
)

(:action pick
      (:parameters ?I - item ?L - location)
      (:precondition (and (not busy) (at ?I ?L) (pos ?L)
      (:effects (and busy (not at ?I ?L)
)

(action clean
      (:parameters ?R - room ?V - vaccum)
      (:preconditions (and (dirty ?R) (not exists (?L - location) (at ?V ?L)) (pos ?R)))
      (:effects (not dirty R)
)

(action clean
      (:parameters ?S - stair ?M - mop ?L - location)
      (:preconditions (and (dirty ?S) (not exists (?L - location) (at ?M ?L)) (pos ?L) (adj ?S ?L )))
      (:effects (not dirty S)
)

(:action put
      (:parameters ?I - item ?L - location)
      (:precondition (and ( busy)(pos ?L))
      (:effects (and (not busy)( at ?I ?L)
)

(:action go
      (:parameters ?L - location ?S - stair)
      (:precondition (and (pos ?L) (or (adj ?L ?S)(adj ?S ?L))))
      (:effects (and (not pos ?L) (pos ?S) (dirt ?S)))
)
```

```
(:action  go
    (:parameters  ?L - location  ?R - room )
    (:precondition  (and (pos ?L) (or (adj ?L ?R)(adj ?R ?L))))
    (:effects (and (not pos ?L) (pos ?R)
)
```

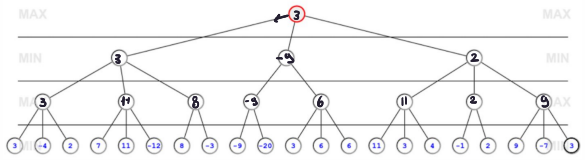
Problem File

```
(:objects     v - vacuum
              m - mop
              attic f21 storage f10 bedroom f12  f00 f01 kitchen - Room
              s0  s1 - stairs
)

(:init  (dirty attic)(dirty bedroom)(dirty kitchen)(pos f01)(dirty s0)(dirty s1)(at v f10)
        (at m f21)(adj s0 f12)(adj s1 attic)(adj kitchen f01)(adj f01 f00)(adj f00 s0)(adj f12 bedroom)
        (adj bedroom f10)(adj f12 s1)(adj attic f21)(adj f21 storage)
)

(:goal  (at v storage)(at m storage)(Forall (?L - location)(not dirty ?L)))
```

A plan is :

go from f01 to f10 → pick v → go to bedroom → clean → go to kitchen → clean → go to attic → clean → go to storage → put v → go to f21 → pick m → go to f12 → clean s0 → go to attic → clean s1 → go to storage → put m.

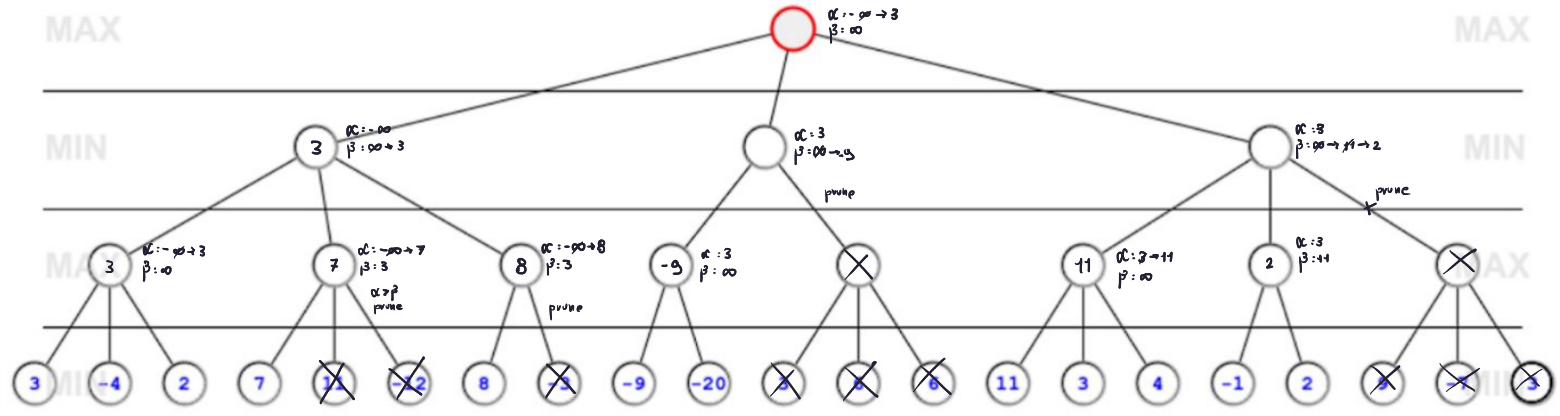
Consider the game tree within the two players Max and Min depicted in the figure. Max is the next to move.



a) Run the Minimax algorithm. What is Max's next move? left branch

b) Run the Alpha-beta pruning algorithm. Show at every internal node the alpha and beta values and how they change throughout the algorithm.

c) What are the properties of Minimax in terms of complexity, completeness, and optimality? How much does the search improve when using the Alpha-beta pruning technique (in the best case)?



In terms of optimality, the minmax ALWAYS find the optimal solution if we assume that the opponent plays optimally as well. Since it expands a tree, the complexity is $O(b^l)$, where b is the branching factor and l the depth. In the optimal case of α-β pruning, we cut in half the branching factor: $O(\lceil b/2 \rceil^l)$.

1. All printers make noise during use. Anyone who has any staplers will not have any tape dispensers. Managers do not have anything which makes noise during use. Alex has either a stapler or a printer.
   a) Translate the sentences above in **FOL (First Order Logic)**. Define a vocabulary for the constants, predicates, and functions you need.
   b) Translate the sentences in CNF (Conjunctive Normal Form).
   c) Prove that: If Alex is a manager, then Alex does not have any tape dispensers.
2. Given the following **FOL** formulas:
   - $\forall x \exists y \forall z (P(x,y) \rightarrow Q(z,x))$
   - $\exists x \forall y \exists z (R(y,z) \wedge \neg S(x,y))$
   - $\forall x \forall y \exists z \exists w (P(x,y,z) \vee Q(w,y))$
   - $\exists x \exists y \forall z (P(z) \rightarrow \exists w Q(x,y,w))$

   a) Write their Skolemized forms.
   b) Why is the Skolem normal form useful?

(1)

1) $\forall x [(Printer(x)) \rightarrow Noisy(x)]$

2) $\forall y \forall x [(Have(y,x) \wedge Stalper(x)) \rightarrow \neg \exists z (Have(y,z) \wedge Dispenser(z))]$

3) $\forall x \forall y [(Manager(y) \wedge Have(y,x)) \rightarrow \neg Noisy(x)]$

4) $\exists x [Have(alex,x) \wedge (Stalper(x) \vee Printer(x))]$

Now i transform in CNF

1) $\forall x [(Printer(x) \wedge Use(x)) \rightarrow Noisy(x)]$:

$\forall x [\neg Printer(x) \vee \neg Use(x) \vee Noisy(x)]$: $\{\neg Printer(x_1), \neg Use(x_1), Noisy(x_1)\}_1$

2) $\forall y \forall x [(Have(y,x) \wedge Stalper(x)) \rightarrow \neg \exists z (Have(y,z) \wedge Dispenser(z))]$:

$\forall y \forall x [\neg Have(y,x) \vee \neg Stalper(x) \vee \forall z (\neg Have(y,z) \vee \neg Dispenser(z))]$

$\{\neg Have(y_2,x_2), \neg Stalper(x_2), \neg Have(y_2,z_2), \neg Dispenser(z_2)\}_2$

3) $\forall x \forall y [(Manager(y) \wedge Have(y,x) \wedge Use(x)) \rightarrow \neg Noisy(x)]$:

$\{\neg Manager(y_3), \neg Have(y_3,x_3), \neg Use(x_3), \neg Noisy(x_3)\}_3$

4) $\exists x [Have(alex,x) \wedge (Stalper(x) \vee Printer(x))]$

$Have(alex,5) \wedge (stampler(5) \vee Printer(5))$:

$\{Have(alex,5)\}_{4A}$ $\{stampler(5), Printer(5)\}_{4B}$

We want to prove :

$Manager(alex) \rightarrow \neg \exists x [Have(alex,x) \wedge Dispenser(x)]$:

$\neg Manager(alex) \vee \forall x [\neg Have(alex,x) \vee \neg Dispenser(x)]$:

$\forall x [\neg Manager(alex) \vee \neg Have(alex,x) \vee \neg Dispenser(x)]$ negated became :

$\exists x [Manager(alex) \wedge Have(alex,x) \wedge Dispenser(x)]$:

$\{Manager(alex)\}_{5A}$ $\{Have(alex,9)\}_{5B}$ $\{Dispenser(9)\}_{5C}$

We add these to the KB and try to derive $\{\square\}$.

$\{\neg Printer(x_1), Noisy(x_1)\}_1$

$\{\neg Have(y_2, x_2), \neg Stalper(x_2), \neg Have(y_2, z_2), \neg Dispenser(z_2)\}_2$

$\{\neg Manager(y_3), \neg Have(y_3, x_3), \neg Noisy(x_3)\}_3$

$\{Have(alex, S)\}_{4A}$

$\{stampler(S), Printer(S)\}_{4B}$

$\{Manager(alex)\}_{5A}$

$\{Have(alex, g)\}_{5B}$

$\{Dispenser(g)\}_{5C}$

From 1 and 3 with $s: \{\frac{x_1}{x_3}\}$:

$\{\neg Printer(x_6), \neg Manager(y_6), \neg Have(y_6, x_6)\}_6 \Rightarrow$ Managers dont have printers

From 6 and 5A $\Rightarrow \{\neg Printer(x_7), \neg Have(alex, x_7)\}_7 \Rightarrow$ Alex dont have a printer

$\{Have(alex, S)\}_{4A}$ with $7 \Rightarrow \{\neg Printer(S)\}$
$\{stampler(S), Printer(S)\}_{4B}$
$\Rightarrow \{Have(alex, S)\}_{4A}$, $\{Stampler(S)\}_8$ Alex have a stampler

$\{\neg Have(y_2, x_2), \neg Stalper(x_2), \neg Have(y_2, z_2), \neg Dispenser(z_2)\}_2$

$\{Have(alex, S)\}_{4A}$

with $s: \{\frac{y_2}{alex}, \frac{x_2}{S}\} \Rightarrow \{\neg Stalper(S), \neg Have(alex, z_2), \neg Dispenser(z_2)\}_2$, $\{Stampler(S)\}_8$ $\Rightarrow$

$\Rightarrow \{\neg Have(alex, z_9), \neg Dispenser(z_9)\}_9$, $\{Have(alex, g)\}_{5B}$ with $\{\frac{z_9}{g}\} \Rightarrow \{\neg Dispenser(g)\}_{10}$

$\{\neg Dispenser(g)\}_{10} \Rightarrow \{\square\} \Rightarrow$ The statement is true.
$\{Dispenser(g)\}_{5C}$

② ▸ $\forall x \exists y \forall z (P(x, y) \rightarrow Q(z, x)) \equiv$

$\forall x \forall z (P(x, S(x)) \rightarrow Q(z, x))$

▸ $\exists x \forall y \exists z (R(y, z) \wedge \neg S(x, y))$

$\forall y (R(y, S(y)) \wedge \neg S(p, y))$

▸ $\forall x \forall y \exists z \exists w (P(x, y, z) \vee Q(w, y))$

$\forall x \forall y \quad (P(x, y, S(x, y)) \vee Q(p(x, y), y))$

- $\exists x \exists y \forall z \, (P(z) \rightarrow \exists w \, Q(x, y, w))$

  $\exists y \forall z \exists w \, (P(z) \rightarrow Q(s, y, w))$

  $\forall z \, (P(z) \rightarrow Q(s, p, g(z)))$

The skolem normal form is important to prove the correctness of resolution, and is also important to resolve a KB since we require all the formulas to be CNF, and to bring a formula to CNF we first have to transform it in skolem normal form by deleting the esistential quantifier and bringing all the quantifier on the left of the formula.

Consider the following constraint network: $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d, e, f\}$
- Domains: for all $v \in V, D_v = \{1, 2, 3, 4, 5\}$
- Constraints: $a = b - 2$, $c = a + 3$, $e = c + 1$, $b = 2f + 1$, $d = b - 1$, $c = b + 1$

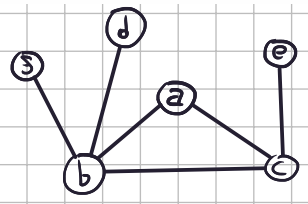
a) Draw the constraint graph γ.

b) Run the **AC-3** algorithm on the given constraint network. For each iteration, give the content of the data structure M at the start of the iteration, the pair (u, v) removed from M, the domain D_u of u after the call to Revise (γ, u, v), and the pairs (w, u) added into M. Note: Initialize M as a lexicographically ordered list (i.e., (a, b) would be before (a, c), both before (b, a) etc., if any of those exist). After initialization, use M as a FIFO queue, i.e., always remove the first (oldest) element from the queue and add new elements to the end of the queue.

c) Give the complete assignment of the variables.

d) Is it possible to apply the AyclicCG algorithm on γ to find a solution? If yes, please explain why. If not, please explain why.

a)



b) We pick always the First of M

1) $M = \{ab, ac, ba, bc, bd, bf, ca, cb, ce, db, ec, fb\}$
pop $ab \Rightarrow D_a = \{1, 2, 3\}$

2) pop $ac \Rightarrow D_a = \{1, 2\}$

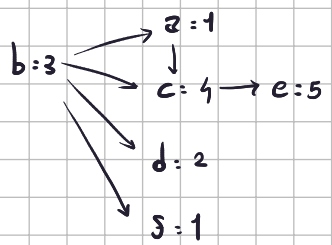
3) pop $ba \Rightarrow D_b = \{3, 4\}$. Add ab to M

4) $M = \{bc, bd, bf, ca, cb, ce, db, ec, fb, ab\}$
pop bc

5) pop bd

6) pop bf, $D_b = \{3\}$

Since the domain are integers, For each constraint C_{uv}, each value of u is paired with exactly one value of v, so, given $D_b = \{3\}$, we can direcly Find the unique solution:



It is not possible to apply AcyclicCG since it requires the constraint graph to be acyclic.