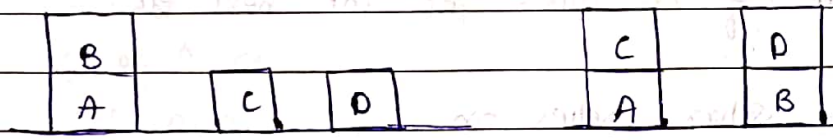
Date ___ / ___ / _____

Assignment - B3

* **Title:-** Goal stack Planning

* **Problem statement:-** Implement goal stack planning for the following configuration from the blocks world.



start End.

* **Objectives:-**
  - To learn and understand concept of goal stack planning.
  - To study need and real time use of goal stack planning.
  - To implement goal stack planning algorithm using suitable programming language.

* **Outcomes:-**

  Students will be able to:-
  - learn the concept of goal stack planning.
  - study need and use of goal stack planning.
  - Implement goal stack planning.

* **S/W and H/W Requirements:-**
  - Ubuntu / Fedora 20.
  - 4GB RAM, 500GB
  - Java JDK / Python libraries / Prolog.
  - Editor - gedit.

* **Theory:-**

One of the earliest techniques in planning using goal stack. Problem

Date __/__/____

Solver uses single stack that contains :-

- subgoals and operates both.
- subgoals are solved linearly and then finally the conjoined
- subgoal is solved.

Plans generated by this method will contain complete sequence of opportunities for solving one goal followed by complete sequence of operations for the next etc.

Problem solver relies on :-

- A database that describes the current situation.
- set of operators with pre-condition, add & delete lists.

⇒ Let us assume that goal to be satisfied is :-

$$GOAL = G1 + G2 + G3 + \dots + GN$$

subgoals G1, G2, G3 are stacked with compound goal $G1 * G2 * G3 \wedge G4 \wedge \dots GN$ at the bottom.

* Algorithm

1) Find an operator that satisfies subgoal G1 (makes it true) and replace G1 by the operator.

   1) If more than one operator satisfies subgoals then apply some heuristic to choose one.

2) In order to execute the top most operation, its preconditions are added into the stack.

   1) once the precondition of an operator are satisfied then we guaranteed that operator can be applied to produce a new state.

   2) new state is obtained by using ADD and DELETE

lists of an operator to existing database.

3) Problem solver keeps track of operations applied.
   ⇒ This process is continued till the goal stack is empty & solver returns plan of the problem.

Consider given example:

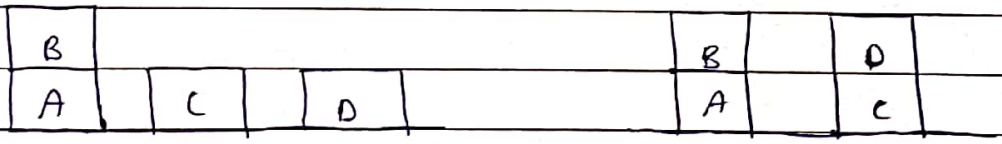
Initial state:-

ON(B,A) ∧ ONT(C) ∧ ONT(A) ∧ ONT(D) ∧ CL(B) ∧ CL(C) ∧ CL(D) ∧ AE.

Goal state:-

ON(C,A) ∧ ON(B,D) ∧ ONT(A) ∧ ONT(B) ∧ CL(C) ∧ CL(B) ∧ AE.

* TestCase:-



Input Output

* Conclusion:-

We successfully implemented goal stack planning in python to implement above case.