GROUP B
Assignment no.13

1. Problem Statement:

Develop a POP for scheduling your higher studies exam. Assume suitable data like college submission schedule, college exams, Constraint that a paper publication is must to appear before the exam, a family function at home and so on.

2. Objective:

1. To apply algorithmic strategies while solving problems.
2. To develop problem solving abilities for gamifications.
3. To develop time and space efficient algorithms.

3. Theory

Partial-Order Planning:

The idea of a **partial-order planner** is to have a partial ordering between actions and only commit to an ordering between actions when forced. This is sometimes also called a **non-linear planner**, which is a misnomer because such planners often produce a linear plan.

For uniformity, treat start as an action that achieves the relations that are true in the initial state, and treat finish as an action whose precondition is the goal to be solved. The pseudoaction start is before every other action, and finish is after every other action. The use of these as actions means that the algorithm does not require special cases for the initial situation and for the goals. When the preconditions of finish hold, the goal is solved.

An action, other than start or finish, will be in a partial-order plan to achieve a precondition of an action in the plan. Each precondition of an action in the plan is either true in the initial state, and so achieved by start, or there will be an action in the plan that achieves it.

We must ensure that the actions achieve the conditions they were assigned to achieve. Each precondition P of an action act_1 in a plan will have an action act_0 associated with it such that act_0 achieves precondition P for act_1. The triple $\langle act_0,P,act_1\rangle$ is a **causal link**. The partial order specifies that action act_0 occurs before action act_1, which is written as $act_0 < act_1$. Any other action A that makes P false must either be before act_0 or after act_1.

Informally, a partial-order planner works as follows: Begin with the actions start and finish and the partial order start < finish. The planner maintains an agenda that is a set of $\langle P,A\rangle$ pairs, where A is an action in the plan and P is an atom that is a precondition of A that must be achieved. Initially the agenda contains pairs $\langle G,finish\rangle$, where G is an atom that must be true in the goal state.

At each stage in the planning process, a pair $\langle G,act_1\rangle$ is selected from the agenda, where P is a precondition for action act_1. Then an action, act_0, is chosen to achieve P. That action is either already in the plan - it could be the start action, for example - or it is a new action that is added to the plan. Action act_0 must happen before act_1 in the partial order. It adds a causal link that records that act_0 achieves P for action act_1. Any action in the plan that deletes P must happen either before act_0 or after act_1. If act_0 is a new action, its preconditions are added to the agenda, and the process continues until the agenda is empty.

This is a non-deterministic procedure. The "choose" and the "either ...or ..." form choices that must be searched over. There are two choices that require search:

- which action is selected to achieve G and
- whether an action that deletes G happens before act_0 or after act_1.

Example

- Goal: Set the table, i.e., on(Tablecloth) ^ out(Glasses) ^ out(Plates) ^ out(Silverware)
- Initial State: clear(Table)
- Operators:
  1. Lay-tablecloth

     P: clear(Table)
     E: on(Tablecloth), ~clear(Table)

  2. Put-out(x)
     P: none
     E: out(x), ~clear(Table)

- Searching for a Solution in Plan Space
  1. Initial Plan
     Start -----------> Finish
  2. Solve 4 unsolved goals in Finish by adding 4 new steps with the minimal temporal constraints possible:

```
                              on(Tablecloth)
Start ------> S1: Lay-tablecloth ------------------------->Finish
    \ \ \                     out(Glasses)    ^ ^ ^
     \ \ \----> S2: Put-out(Glasses) -----------------| | |
      \ \                     out(Plates)     /  /
       \ \-----> S3: Put-out(Plates) ------------------/  /
        \                        out(Silverware) /
         \------> S4: Put-out(Silverware) ---------------/
```

  3. Solve unsolved subgoal clear(Table) which is a precondition of step S1:

```
         clear(Table)                   on(Tablecloth)
Start -----------> S1: Lay-tablecloth ----------------------->Finish
    \ \ \                          out(Glasses)    ^ ^ ^
     \ \ \---------> S2: Put-out(Glasses) ----------------| | |
      \ \                          out(Plates)      | |
       \ \----------> S3: Put-out(Plates) ------------------/  |
        \                             out(Silverware) /
         \-----------> S4: Put-out(Silverware) ---------------/
```

  4. Fix threats caused by steps S2, S3, and S4 on the link from Start to S1. That is, clear(Table) is a necessary precondition of S1 that is created by step Start. But S2 causes clear(Table) to be deleted (negated), so if S2 came before S1, clear(Table) wouldn't be true and step S1 couldn't be performed. Therefore, add a temporal constraint that forces S2 to come anytime after S1. That is, add constraint S1 < S2. Similarly, add S1 < S3, and S1 < S4, resulting in the new plan:

```
     clear(Table)                  on(Tablecloth)
Start -----------> S1: Lay-tablecloth ---------------------->Finish
| | |            |\--|---|                           ^ ^ ^
| | |            |   |   v               out(Glasses)  | | |
| | |--------------+---+-> S2: Put-out(Glasses) --------------/ | |
| |              |   v                 out(Plates)       / |
| |----------------+-> S3: Put-out(Plates) --------------------/  |
```

```
|                v                         out(Silverware)    /
|---------------> S4: Put-out(Silverware) ----------------------/
```

5. No threats and no unsolved goals in this plan, so it is a **complete plan** (i.e., a solution to the planning problem). Any total ordering of the steps implied by this partial-order plan is a solution plan. Here, there are six possible plans, where the first step is S1, and the steps S2, S3, and S4 follow in any order. (Don't include the pseudo-steps Start and Finish.)

4. Algorithm:

1. on-deterministic procedure PartialOrderPlanner(Gs)
2. Inputs
   Gs: set of atomic propositions to achieve
3. Output
   linear plan to achieve Gs
4. Local: Agenda: set of ⟨P,A⟩ pairs where P is atom and A an action
   Actions: set of actions in the current plan
   Constraints: set of temporal constraints on actions
   CausalLinks: set of $\langle act_0,P,act_1\rangle$ triples
5. Agenda ←{⟨G,finish⟩:G ∈Gs}
   Actions ←{start,finish}
   Constraints ←{start<finish}
   CausalLinks ←{}
   repeat
6. select and remove $\langle G,act_1\rangle$ from Agenda
   Either
   choose $act_0 \in$Actions such that act_0 achieves G
   or
   choose $act_0 \notin$Actions such that act_0 achieves G
   Actions ←Actions ∪{act_0}
   Constraints ←add_const(start<act_0,Constraints)
7. for each CL∈CausalLinks do
   Constraints ←protect(CL,act_0,Constraints)

8. Agenda ←Agenda ∪{⟨P,act_0⟩: P is a precondition of act_0 }

9. Constraints ←add_const(act_0<act_1,Constraints)
   CausalLinks ∪ {⟨act_o,G,act_1⟩}
   for each A∈Actions do
   Constraints ←protect(⟨act_o,G,act_1⟩,A,Constraints)
10. until Agenda={}
   return total ordering of Actions consistent with ConstraintsInput for assignment:

User selection- O’s placement

5. Expected Output:

Solution in Plan Space

6. Math Model:

Let S be the Solution of the Problem-

S={s, e, i, o}

Where,
s=initial state
e=end state
i=input
o=output

s=initialize the space plan
s= {start,finish,goal}

start={clear the space_object}
finish={Final state}

goal={ Set the table}

i={space_plan,start}

o={Partial Order plan}

e= The system will be end in following conditions:
(i) If incorrect input is provided
(ii) System failure

7. Test Case(Testing):

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	Invalid input	Should not accept	Not accepting	Pass
2	Valid goal state	Should accept and return POP	Accepting goal state	Pass
3	Valid Start value	Should start searching for space plan	Searching for space plan	Pass

8. Conclusion:
Hence, we have implemented the Partial Order Planning for scheduling higher studies.

9. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	**ACHIEVED**(√)
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	√