INTERNAL ASSESMENT

ARTIFICIAL INTELLIGENCE

K.R. MANGALAM UNIVERSITY

SUBJECT CODE:-ETCS451A

SUBMITTED TO:

DR. HARSH VARDHAN

ASSISTANT PROFESSOR

SUBMITTED BY:

Aanshu Tanwar

2201010169

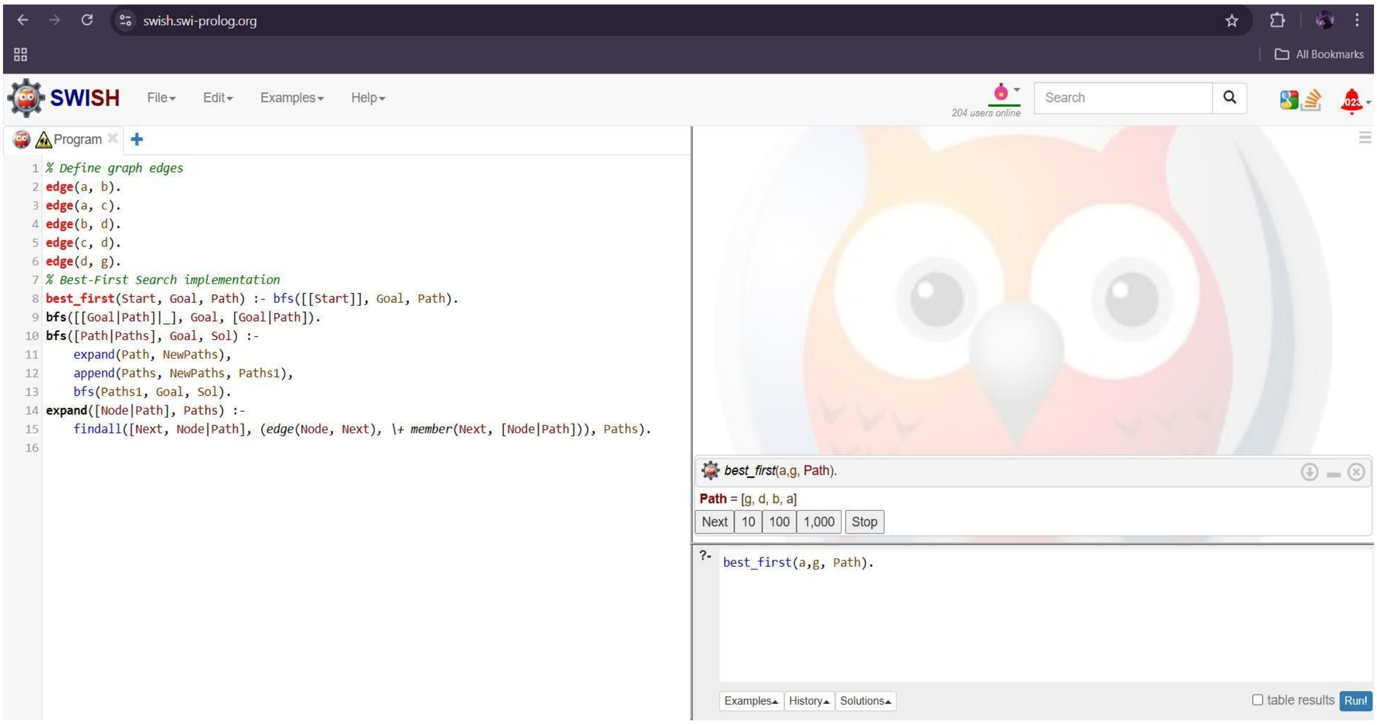
B.TECH CSE-6Th Sem
Section - C

PROGRAM 1: Solve any problem using best first search in Prolog

CODE:

```
% Define graph edges

edge(a, b). edge(a, c).

edge(b, d). edge(c, d).

edge(d, g).

% Best-First Search implementation best_first(Start,

Goal, Path) :- bfs([[Start]], Goal, Path).

bfs([[Goal|Path]|_], Goal, [Goal|Path]).

bfs([Path|Paths], Goal, Sol) :-      expand(Path,

NewPaths),     append(Paths, NewPaths, Paths1),

bfs(Paths1, Goal, Sol).

expand([Node|Path], Paths) :-      findall([Next, Node|Path], (edge(Node, Next), \+

member(Next, [Node|Path])), Paths).
```

OUTPUT:



```
% Define graph edges
edge(a, b).
edge(a, c).
edge(b, d).
edge(c, d).
edge(d, g).
% Best-First Search implementation
best_first(Start, Goal, Path) :- bfs([[Start]], Goal, Path).
bfs([[Goal|Path]|_], Goal, [Goal|Path]).
bfs([Path|Paths], Goal, Sol) :-
    expand(Path, NewPaths),
    append(Paths, NewPaths, Paths1),
    bfs(Paths1, Goal, Sol).
expand([Node|Path], Paths) :-
    findall([Next, Node|Path], (edge(Node, Next), \+ member(Next, [Node|Path])), Paths).
```

```
best_first(a,g, Path).
Path = [g, d, b, a]
```

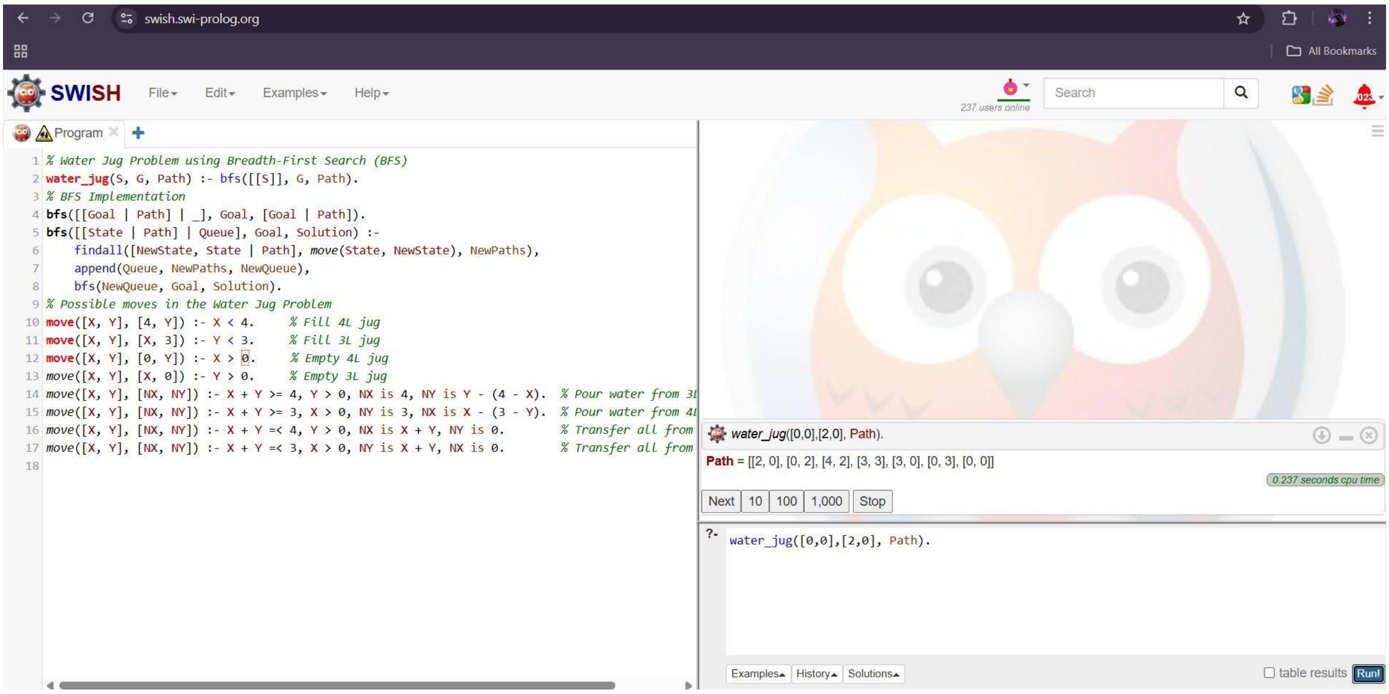
```
?- best_first(a,g, Path).
```

PROGRAM 8: Write a Program to Implement Water-Jug problem

CODE:

```
% Water Jug Problem using Breadth-First Search (BFS) water_jug(S,
G, Path) :- bfs([[S]], G, Path).
% BFS Implementation
bfs([[Goal | Path] | _], Goal, [Goal | Path]). bfs([[State | Path] | Queue],
Goal, Solution) :-     findall([NewState, State | Path], move(State,
NewState), NewPaths),     append(Queue, NewPaths, NewQueue),
bfs(NewQueue, Goal, Solution).
% Possible moves in the Water Jug Problem move([X, Y], [4, Y]) :- X < 4.     % Fill 4L jug move([X, Y],
[X, 3]) :- Y < 3.     % Fill 3L jug move([X, Y], [0, Y]) :- X > 0.     % Empty 4L jug move([X, Y], [X, 0]) :-
Y > 0.     % Empty 3L jug move([X, Y], [NX, NY]) :- X + Y >= 4, Y > 0, NX is 4, NY is Y - (4 - X).  %
Pour water from 3L to 4L jug move([X, Y], [NX, NY]) :- X + Y >= 3, X > 0, NY is 3, NX is X - (3 - Y).  %
Pour water from 4L to 3L jug move([X, Y], [NX, NY]) :- X + Y =< 4, Y > 0, NX is X + Y, NY is 0.       %
Transfer all from 3L to 4L move([X, Y], [NX, NY]) :- X + Y =< 3, X > 0, NY is X + Y, NX is 0.       %
Transfer all from 4L to 3L
```

OUTPUT:



```
% Water Jug Problem using Breadth-First Search (BFS)
water_jug(S, G, Path) :- bfs([[S]], G, Path).
% BFS Implementation
bfs([[Goal | Path] | _], Goal, [Goal | Path]).
bfs([[State | Path] | Queue], Goal, Solution) :-
    findall([NewState, State | Path], move(State, NewState), NewPaths),
    append(Queue, NewPaths, NewQueue),
    bfs(NewQueue, Goal, Solution).
% Possible moves in the Water Jug Problem
move([X, Y], [4, Y]) :- X < 4.      % Fill 4L jug
move([X, Y], [X, 3]) :- Y < 3.      % Fill 3L jug
move([X, Y], [0, Y]) :- X > 0.      % Empty 4L jug
move([X, Y], [X, 0]) :- Y > 0.      % Empty 3L jug
move([X, Y], [NX, NY]) :- X + Y >= 4, Y > 0, NX is 4, NY is Y - (4 - X).  % Pour water from 3L
move([X, Y], [NX, NY]) :- X + Y >= 3, X > 0, NY is 3, NX is X - (3 - Y).  % Pour water from 4L
move([X, Y], [NX, NY]) :- X + Y =< 4, Y > 0, NX is X + Y, NY is 0.        % Transfer all from
move([X, Y], [NX, NY]) :- X + Y =< 3, X > 0, NY is X + Y, NX is 0.        % Transfer all from
```

```
water_jug([0,0],[2,0], Path).
Path = [[2, 0], [0, 2], [4, 2], [3, 3], [3, 0], [0, 3], [0, 0]]
0.237 seconds cpu time
```

```
?- water_jug([0,0],[2,0], Path).
```