DHAKA UNIVERSITY OF ENGINEERING & TECHNOLOGY, GAZIPUR

Department of Computer Science and Engineering

Course No.: CSE-4624

Course Title: Artificial Intelligence Sessional

Submission Date:08-03-24 .

Submitted To:

Dr.Fazlul Hasan Siddiqui
Professor Dept. of CSE (DUET)

Sabah Binte Noor
Assistant Professor Dept. of CSE (DUET)

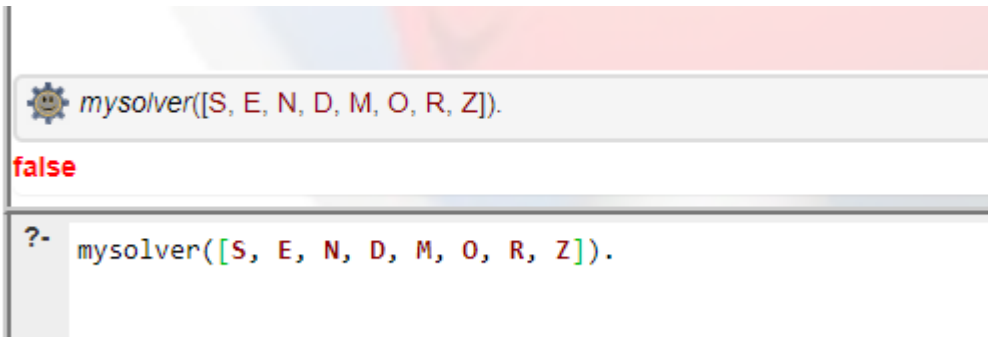
Submitted By:

Md.Didar Ahmed
Id. 194048
Year: 4th
Semester: 2nd
Session: 2023 - 2024

Problem-1: Solution

```
% for check there is no exit similar data

allDistinct([]).

%This syntax represents a list with a head H and a tail T.

allDistinct([H|T]) :- \+ member(H, T), allDistinct(T).


mysolver([S, E, N, D, M, O, R, Y]) :-

Digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

member(S, Digits), S > 0,

member(M, Digits), M > 0, M \= S,

member(E, Digits),

member(N, Digits),

member(D, Digits),

member(O, Digits),

member(R, Digits),

member(Y, Digits),

allDistinct([S, E, N, D, M, O, R, Y]),

S * 1000 + E * 100 + N * 10 + D +

M * 1000 + O * 100 + R * 10 + E =

M * 10000 + O * 1000 + N * 100 + E * 10 + Y.
```

Output:



Problem-2: Solution

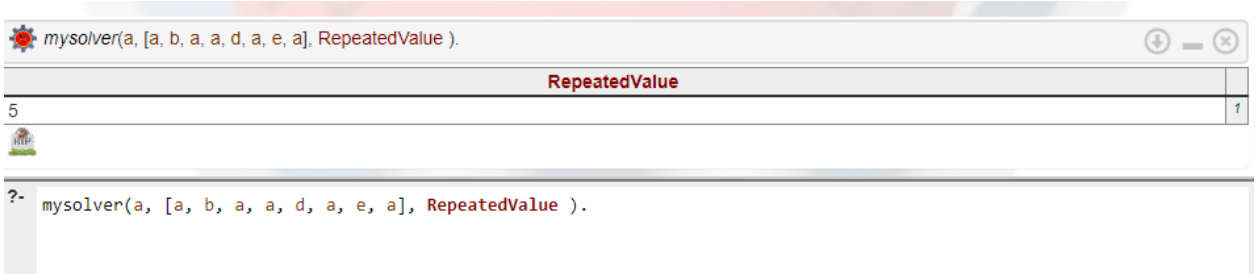
```
% Base case: an empty list has no duplicates
mysolver([], []).

% Recursive case: remove duplicates from the tail of the list
mysolver([Head | Tail], Result) :-
    member(Head, Tail),  % Check if Head is a duplicate in the Tail
    mysolver(Tail, Result).

mysolver([Head | Tail], [Head | Result]) :-
    \+ member(Head, Tail),  % Head is not a duplicate, include it in the result
    mysolver(Tail, Result).
```

Output:

Problem-3: Solution

```
% Base case: Occurrences of X in an empty list is 0
mysolver(_, [], 0).

% Recursive case: Check the head of the list
mysolver(X, [X | Tail], N) :-
    mysolver(X, Tail, N1),
    N is N1 + 1.

% Recursive case: Skip the head if it's not X
mysolver(X, [Y | Tail], N) :-
    X \= Y,
    mysolver(X, Tail, N).
```

Output:



Problem-4:Solution

```
% Define the Tower of Hanoi predicate
mysolver(N) :-
    move(N, left, center, right).

% Base case
move(0, _, _, _) :- !.

% move N disks from Source to Target using Auxiliary
move(N, Source, Target, Auxiliary) :-
    M is N - 1,
    move(M, Source, Auxiliary, Target),
    write('Move disk '), write(N),
    write(' from '), write(Source),
    write(' to '), write(Target), nl,
    move(M, Auxiliary, Target, Source).
```

Output: