Write a Prolog program to implement pattern matching

AIM

To write a Prolog program that implements pattern matching, checking whether a given pattern exists within a list of elements.

ALGORITHM

1. Start the program.
2. Define a predicate match(Pattern, List) that checks if the pattern exists in the list.
3. Base case: An empty pattern matches any list.
4. If the head of the list matches the head of the pattern, recursively check the rest of the pattern with the rest of the list.
5. If the head does not match, skip the head of the list and try matching the pattern with the tail.
6. Load the program into the Prolog interpreter.
7. Query with match(Pattern, List) to check for the pattern in the list.
8. Stop.

```
% Pattern matching in lists
% match(Pattern, List) succeeds if Pattern appears in List

% Base case: empty pattern matches anything
match([], _).

% Check if pattern matches from the start of the list
match([P|Ps], [P|Ls]) :-
    match(Ps, Ls).

% Skip first element and try matching pattern again
match(Pattern, [_|Ls]) :-
    match(Pattern, Ls).
```

OUTPUT:

```
?-
% c:/Users/gayathri/Downloads/match.pl compiled 0.00 sec, 3 clauses
?- match([b,c], [a,b,c,d]).
true .

?-
|    match([a,d], [a,b,c,d]).
true .

?-
|     match([x,y], [a,b,c,d]).
false.

?-
```

RESULT

The program successfully checks for patterns within a list. For example, querying match([b,c], [a,b,c,d]) returns true, while match([x,y], [a,b,c,d]) returns false.