1.simple fact

```
% Facts
likes(ram, mango).        % Ram likes mango
girl(seema).              % Seema is a girl
red(rose).                % Rose is red
likes(bill, cindy).       % Bill likes Cindy
owns(john, gold).         % John owns gold
```

2.salesman

```
road(birmingham,bristol, 9).
road(london,birmingham, 3).
road(london,bristol, 6).
road(london,plymouth, 5).
road(plymouth,london, 5).
road(portsmouth,london, 4).
road(portsmouth,plymouth, 8). get_road(Start, End, Visited,
Result) :-get_road(Start, End, [Start], 0, Visited, Result).
get_road(Start, End, Waypoints, DistanceAcc, Visited, TotalDistance) :- road(Start, End, Distance),
reverse([End|Waypoints], Visited), TotalDistance is DistanceAcc + Distance.
get_road(Start, End, Waypoints, DistanceAcc, Visited, TotalDistance) :- road(Start, Waypoint, Distance),
\+ member(Waypoint, Waypoints), NewDistanceAcc is DistanceAcc + Distance,
get_road(Waypoint, End, [Waypoint|Waypoints], NewDistanceAcc, Visited,
TotalDistance).
```

3.lib program

```
% Facts
book('The Hobbit', 'J.R.R. Tolkien', 1937, 2).
book('1984', 'George Orwell', 1949, 3).
book('To Kill a Mockingbird', 'Harper Lee', 1960, 5).
book('Pride and Prejudice', 'Jane Austen', 1813, 4).
book('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 2).

student('Alice').
student('Bob').
student('Charlie').
student('David').
```

```
student('Eva').

taken('The Hobbit', 'Alice', '2023-10-25', '2023-11-10').
taken('1984', 'Bob', '2023-10-20', '2023-11-05').
taken('To Kill a Mockingbird', 'Charlie', '2023-10-22', '2023-11-08').
taken('Pride and Prejudice', 'Eva', '2023-10-21', '2023-11-07').
taken('The Great Gatsby', 'David', '2023-10-23', '2023-11-09').

% Rules
author(Author, Title) :- book(Title, Author, _, _).
in_stock(Title, Stock) :- book(Title, _, _, Stock).

available(Title, Available) :-
    book(Title, _, _, Total),
    findall(Student, taken(Title, Student, _, _), Taken),
    length(Taken, TakenCount),
    Available is Total - TakenCount.
```

4.fah prog

```
% Celsius to Fahrenheit conversion
c_to_f(Celsius, Fahrenheit) :-
    Fahrenheit is (Celsius * 9 / 5) + 32.

% Freezing check
freezing(Fahrenheit) :-
    Fahrenheit =< 32.
```

5.dfs

```
s(a,b).
s(a,c).
s(b,d).
s(b,e).
s(c,f).
s(c,g).
s(d,h).
s(e,i).
s(e,j).
```

```
s(f,k).
goal(f).
goal(j).
mem(X,[X|_]).
mem(X,[_|Tail]):-mem(X,Tail).
solve(Node,Solution):-
dfs([],Node,Solution).
dfs(Path,Node,[Node|Path]):-
goal(Node).
dfs(Path,Node,Sol):-
s(Node,Node1),
not(mem(Node1,Path)),
dfs([Node|Path],Node1,Sol).
```

6.bfs

```
s(a,b).
s(a,c).
s(b,d).
s(b,e).
s(c,f).
s(c,g).
s(d,h).
s(e,i).
s(e,j).
s(f,k).
goal(f).
goal(j).
solve(Start,Solution):-
bfs([[Start]],Solution).
bfs([[Node|Path]|_],[Node|Path]):-
goal(Node).
bfs([Path|Paths],Solution):-
extend(Path,NewPaths),
write(NewPaths),
nl,
conc(Paths,NewPaths,Paths1),bfs(Paths1,Solution).
extend([Node|Path],NewPaths):-
```

```
bagof([NewNode,Node|Path],(s(Node,NewNode),not(member(NewNode,[Node|Path]))),NewPaths),!.
extend(_,[]).
conc([],L,L).
conc([X|L1],L2,[X|L3]):-nl,write('conc'),write(X),write(' '),write(L1),write(L2),conc(L1,L2,L3).
```

7.4 queen

```
% Generate all permutations of a list
perm([X|Y], Z) :- perm(Y, W), takeout(X, Z, W).
perm([], []).

% Takeout predicate that removes the element X from the list
takeout(X, [X|R], R).
takeout(X, [F|R], [F|S]) :- takeout(X, R, S).

% Combine two lists element-wise by adding and subtracting corresponding elements
combine([X1|X], [Y1|Y], [S1|S], [D1|D]) :-
    S1 is X1 + Y1,
    D1 is X1 - Y1,
    combine(X, Y, S, D).
combine([], [], [], []).

% Check if all elements in a list are distinct
all_diff([X|Y]) :- \+ member(X, Y), all_diff(Y).
all_diff([]).  % Allow empty list (base case for recursion)

% Solve the problem
solve(P) :-
    perm([1, 2, 3, 4], P),  % Get all permutations of [1, 2, 3, 4]
    combine([1, 2, 3, 4], P, S, D),  % Generate S and D from the combination
    all_diff(S),  % Ensure all elements in S are distinct
    all_diff(D).  % Ensure all elements in D are distinct
```

8.chatbot

```
# Function to define the chatbot's responses
def chatbot():
    print("Hello! I'm your simple chatbot. Type 'exit' to end the conversation.")

    while True:
```

```
        # Taking user input
        user_input = input("You: ").lower()

        # Check for exit condition
        if user_input == 'exit':
            print("Chatbot: Goodbye! Have a nice day!")
            break

        # Basic responses based on user input
        elif "hello" in user_input or "hi" in user_input:
            print("Chatbot: Hello there! How can I assist you?")

        elif "how are you" in user_input:
            print("Chatbot: I'm just a bot, but I'm doing well! How about you?")

        elif "bye" in user_input:
            print("Chatbot: Bye! Take care!")

        elif "your name" in user_input:
            print("Chatbot: I am a chatbot created by a Python programmer!")

        else:
            print("Chatbot: I'm not sure how to respond to that. Can you ask something else?")

# Run the chatbot
if __name__ == "__main__":
    chatbot()
```