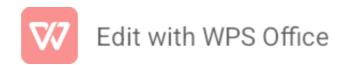
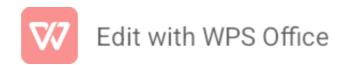
```
19. Student-Teacher-Subject Database
prolog
Copy
Download
student(john, cs101).
student(sarah, cs101).
student(mike, math202).
teacher(dr_smith, cs101).
teacher(dr_jones, math202).
% Query: student(Name, Code), teacher(Teacher, Code).
20. Planets Database
prolog
Copy
Download
planet(mercury, rocky, 0.39).
planet(venus, rocky, 0.72).
planet(earth, rocky, 1.0).
planet(mars, rocky, 1.52).
planet(jupiter, gas_giant, 5.20).
planet(saturn, gas_giant, 9.58).
% Query: planet(Name, Type, Distance).
21. Towers of Hanoi
prolog
Copy
Download
hanoi(1, Start, End, _):-
```



```
write('Move top disk from '), write(Start), write(' to '), write(End), nl.
hanoi(N, Start, End, Via):-
  N > 1,
  M is N - 1,
  hanoi(M, Start, Via, End),
  hanoi(1, Start, End, _),
  hanoi(M, Via, End, Start).
% Query: hanoi(3, left, right, center).
22. Birds That Can Fly
prolog
Copy
Download
can_fly(penguin, no).
can_fly(sparrow, yes).
can_fly(ostrich, no).
can_fly(eagle, yes).
can_fly(kiwi, no).
% Query: can_fly(Bird, yes).
23. Family Tree
prolog
Copy
Download
parent(john, mary).
parent(john, bob).
parent(mary, ann).
parent(mary, tom).
parent(bob, lisa).
```



```
male(john).
male(bob).
male(tom).
female(mary).
female(ann).
female(lisa).
father(Father, Child):-
  parent(Father, Child),
  male(Father).
mother(Mother, Child):-
  parent(Mother, Child),
  female(Mother).
% Query: father(Father, Child).
24. Dieting System Based on Disease
prolog
Copy
Download
diet(diabetes, low_sugar).
diet(hypertension, low_sodium).
diet(obesity, low_calorie).
diet(anemia, iron_rich).
recommend_diet(Disease, Diet) :-
  diet(Disease, Diet).
% Query: recommend_diet(Disease, Diet).
```



25. Monkey Banana Problem

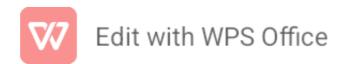
```
prolog
Copy
Download
state(atdoor, onfloor, atwindow, hasnot).
state(atwindow, onfloor, atwindow, hasnot).
state(atwindow, onbox, atwindow, hasnot).
state(atwindow, onbox, atwindow, has).
move(state(middle, onfloor, middle, hasnot), grasp, state(middle, onfloor, middle,
has)).
move(state(P, onfloor, P, H), climb, state(P, onbox, P, H)).
move(state(P1, onfloor, P1, H), push(P1, P2), state(P2, onfloor, P2, H)).
move(state(P1, onfloor, B, H), walk(P1, P2), state(P2, onfloor, B, H)).
canget(state(_, _, _, has)).
canget(State1):-
  move(State1, _, State2),
  canget(State2).
% Query: canget(state(atdoor, onfloor, atwindow, hasnot)).
26. Fruit and Color with Backtracking
prolog
Copy
Download
fruit_color(apple, red).
fruit_color(banana, yellow).
fruit_color(grape, purple).
fruit_color(orange, orange).
```



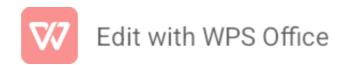
```
fruit_color(apple, green). % Some apples are green
% Query: fruit_color(Fruit, Color).
27. Best First Search
prolog
Copy
Download
% This is a simplified implementation
best_first_search(Start, Goal) :-
  bfs([node(Start, [])], Goal, Path),
  reverse(Path, ReversedPath),
  write('Path: '), write(ReversedPath).
bfs([node(Goal, Path)|_], Goal, [Goal|Path]).
bfs([node(State, Path)|Rest], Goal, Solution):-
  findall(node(NextState, [State|Path]),
      (move(State, NextState), \+ member(NextState, Path)),
      Children),
  append(Rest, Children, NewQueue),
  bfs(NewQueue, Goal, Solution).
% Requires defining move/2 for your specific problem
28. Medical Diagnosis
prolog
Copy
Download
symptom(fever, flu).
symptom(cough, flu).
symptom(fever, cold).
```



```
symptom(sneezing, cold).
symptom(headache, migraine).
symptom(nausea, migraine).
diagnose(Symptoms, Diagnosis):-
  findall(D, (member(S, Symptoms), symptom(S, D)), Diagnoses),
  list_to_set(Diagnoses, PossibleDiagnoses),
  member(Diagnosis, PossibleDiagnoses).
% Query: diagnose([fever, cough], D).
29. Forward Chaining
prolog
Copy
Download
% Knowledge base
rule(has_wings, can_fly).
rule(can_fly, is_bird).
rule(lays_eggs, is_bird).
rule(is_bird, is_animal).
% Forward chaining
forward_chain(Facts, NewFacts) :-
  findall(Conclusion,
      (member(Fact, Facts),
       rule(Fact, Conclusion),
       \+ member(Conclusion, Facts)),
      NewFacts),
  NewFacts \= [].
```



```
infer_all(Facts, AllFacts) :-
  forward_chain(Facts, NewFacts),
  append(Facts, NewFacts, UpdatedFacts),
  infer_all(UpdatedFacts, AllFacts).
infer_all(Facts, Facts).
% Query: infer_all([has_wings], AllFacts).
30. Backward Chaining
prolog
Copy
Download
% Knowledge base
rule(is_bird, [has_wings, can_fly]).
rule(is_bird, [lays_eggs]).
rule(can_fly, [has_wings]).
backward_chain(Goal, KnownFacts, Proof) :-
  member(Goal, KnownFacts),
  Proof = [Goal].
backward_chain(Goal, KnownFacts, [Goal|Subproofs]) :-
  rule(Goal, Subgoals),
  backward_chain_list(Subgoals, KnownFacts, Subproofs).
backward_chain_list([], _, []).
backward_chain_list([H|T], KnownFacts, [HProof|TProof]) :-
  backward_chain(H, KnownFacts, HProof),
  backward_chain_list(T, KnownFacts, TProof).
% Query: backward_chain(is_bird, [has_wings], Proof).
```



```
32. Pattern Matching
prolog
Copy
Download
match([], []).
match([H|T], [H|T2]):- match(T, T2).
match([_|T], [_|T2]) :- match(T, T2).
% Query: match([a,b,c], [a,X,Y]).
33. Count Vowels
prolog
Copy
Download
vowel(a). vowel(e). vowel(i). vowel(o). vowel(u).
count_vowels([], 0).
count_vowels([H|T], Count):-
  (vowel(H) ->
    count_vowels(T, SubCount),
    Count is SubCount + 1
    count_vowels(T, Count)
  ).
% Query: count_vowels("hello world", Count).
```