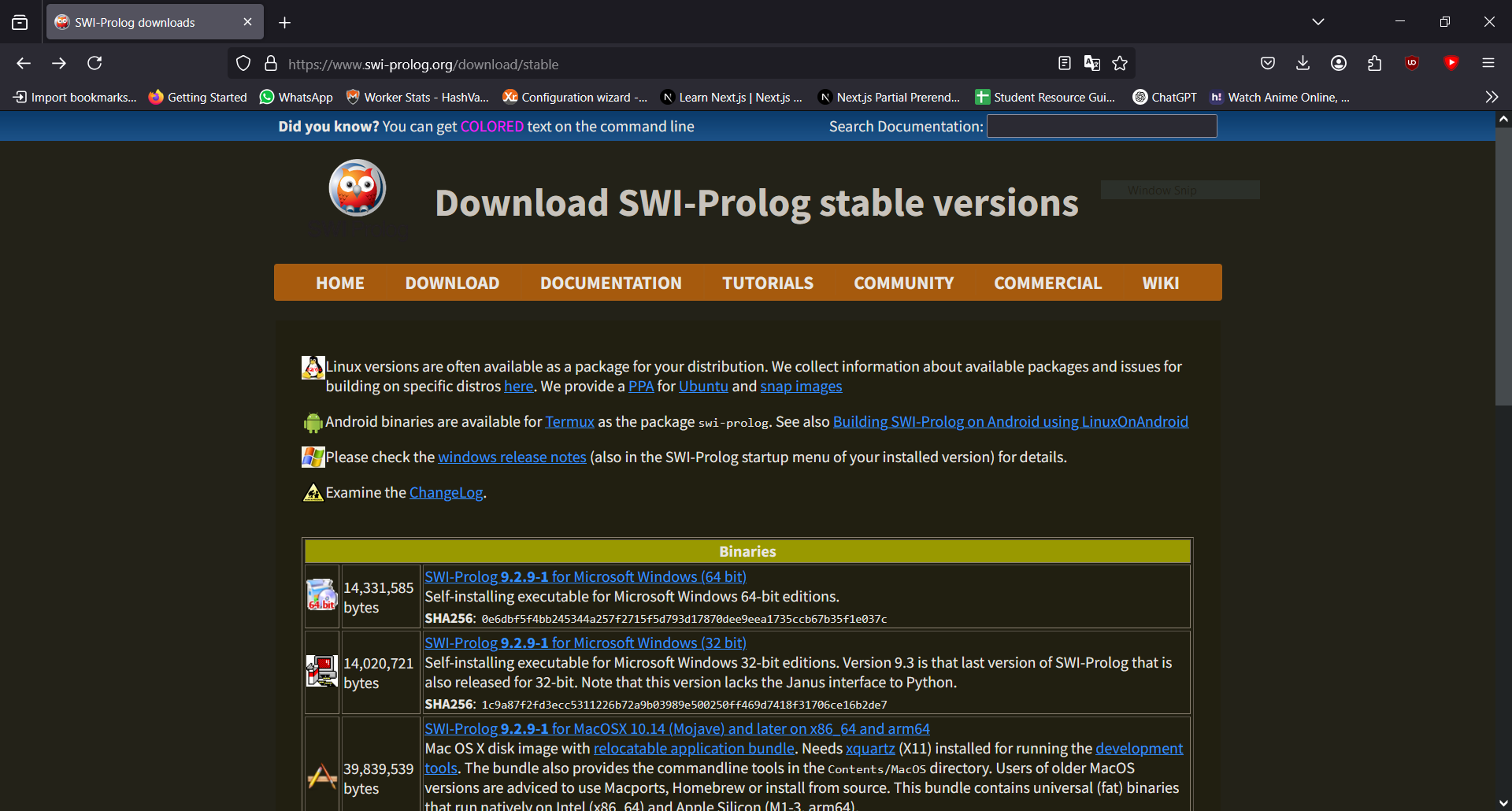
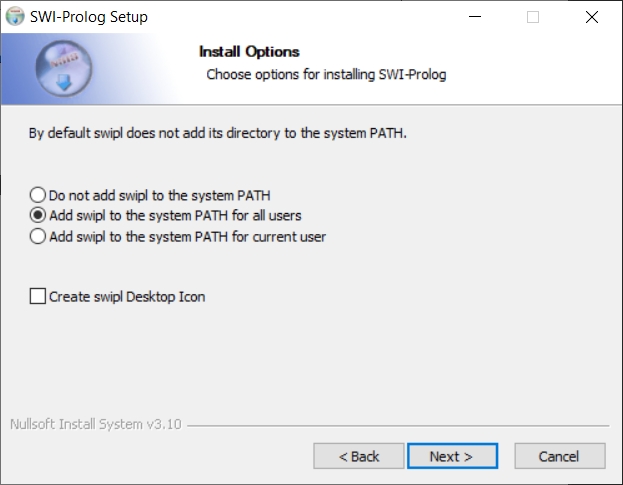
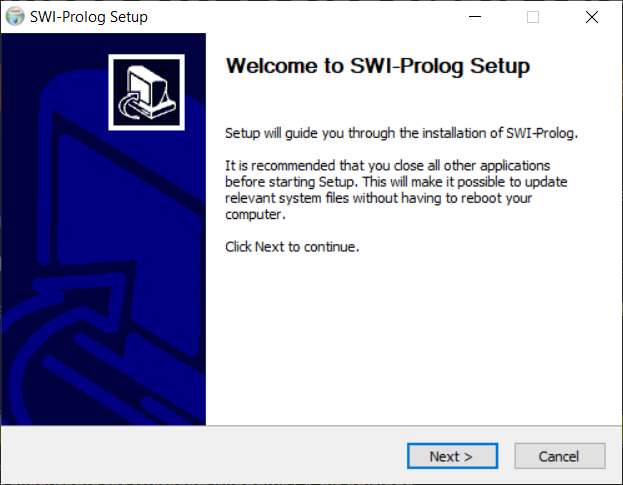
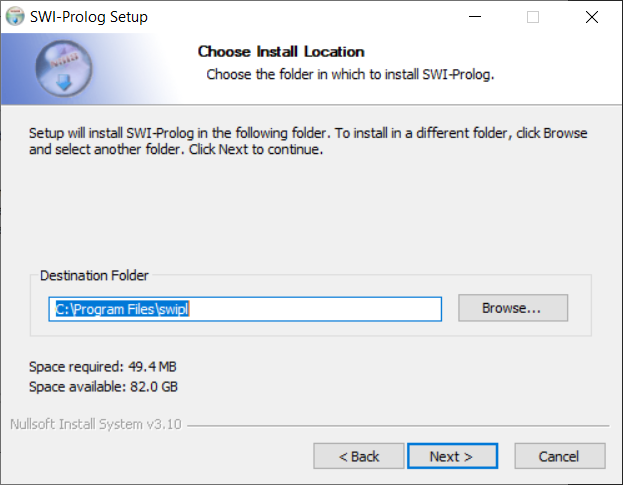
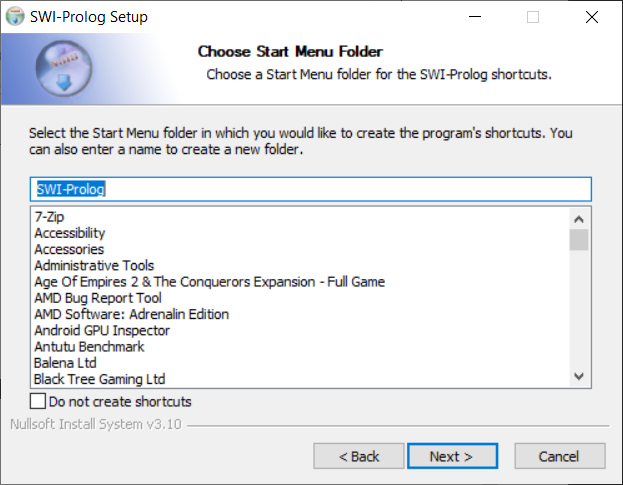
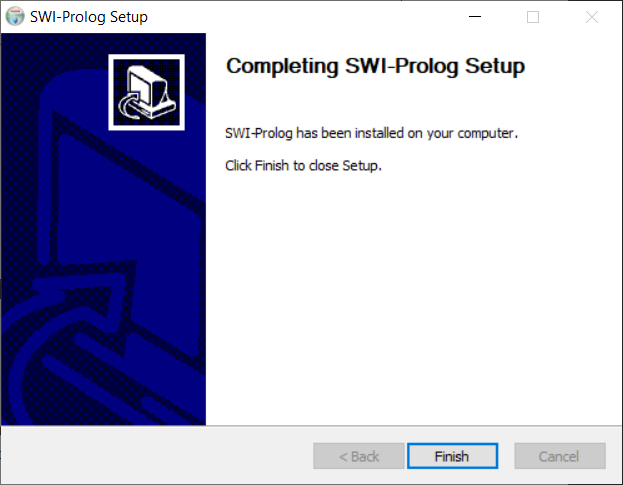
Experiment-1

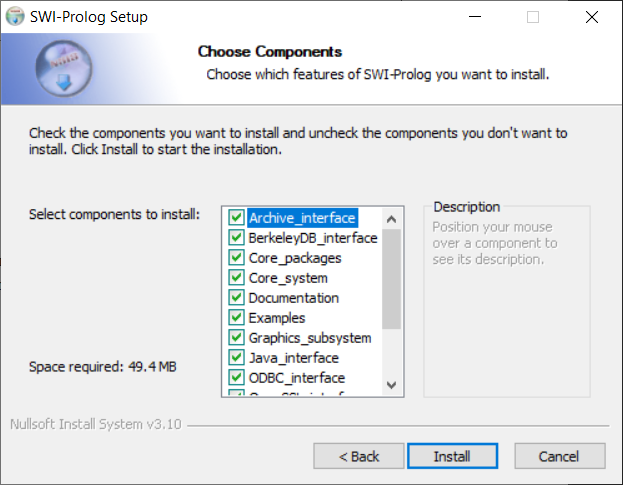


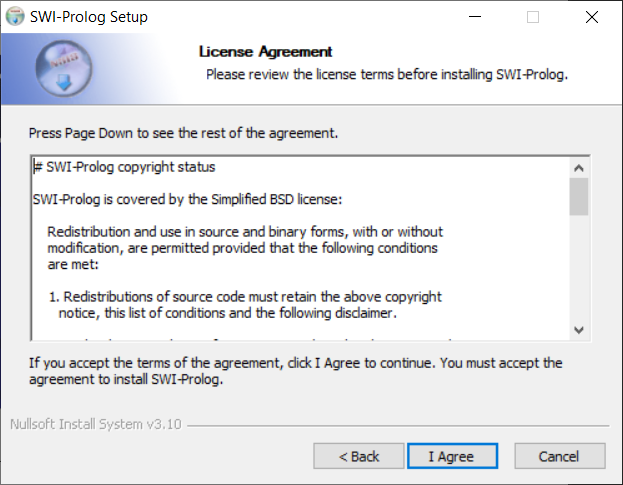


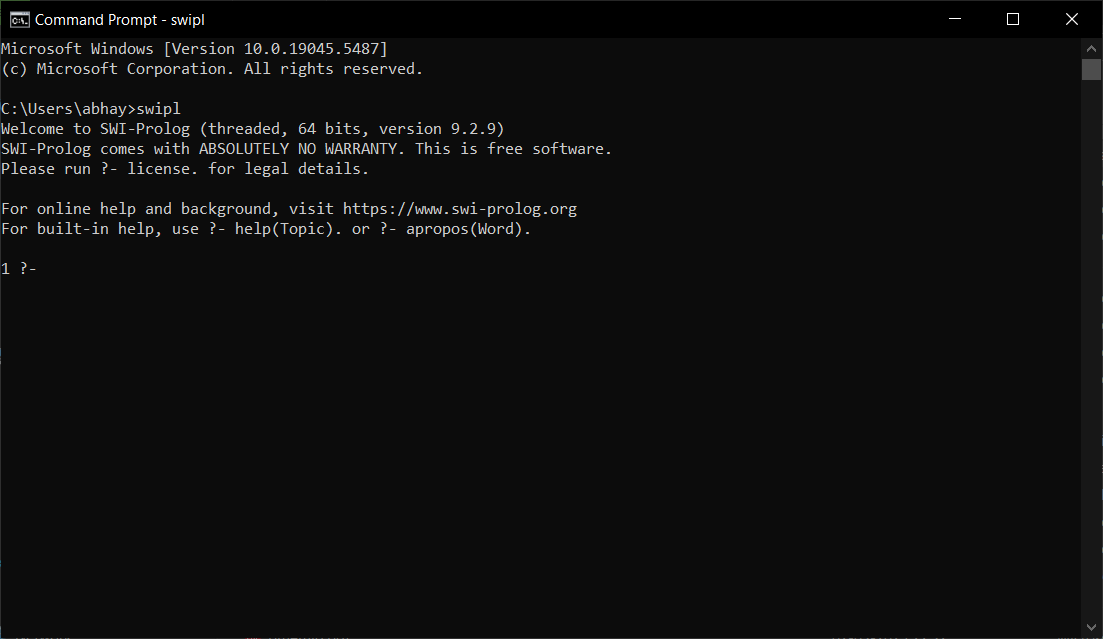






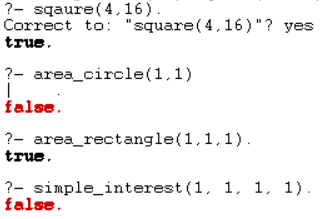






Experiment – 2

Code:-

square(Number, Result) :-

    Result is Number \* Number.

area\_circle(Radius, Area) :-

    Area is 3.14159 \* Radius \* Radius.

area\_square(Side, Area) :-

    Area is Side \* Side.

area\_rectangle(Length, Width, Area) :-

    Area is Length \* Width.

simple\_interest(Principal, Rate, Time, Interest) :-

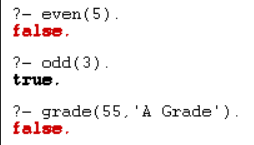
    Interest is (Principal \* Rate \* Time) / 100.

Experiment – 3

Code:-

even(Number) :-

    Number mod 2 =:= 0.

odd(Number) :-

    Number mod 2 =\= 0.

max(X, Y, X) :-

    X >= Y.

max(X, Y, Y) :-

    Y > X.

grade(Marks, 'A Grade') :-

    Marks >= 90.

grade(Marks, 'B Grade') :-

    Marks >= 75,

    Marks < 90.

grade(Marks, 'C Grade') :-

    Marks >= 50,

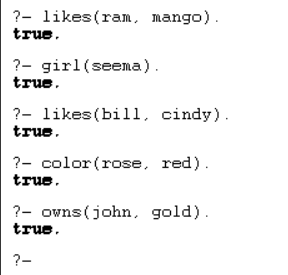
    Marks < 75.

grade(Marks, 'Fail') :-

    Marks < 50.

Output:-

Experiment – 4

Code:-

% Facts

likes(ram, mango).

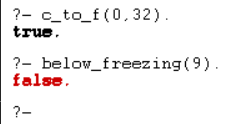
girl(seema).

likes(bill, cindy).

color(rose, red).

owns(john, gold).

Output:-

Experiment – 5

c\_to\_f(C, F) :-

    F is (C \* 9 / 5) + 32.

below\_freezing(C) :-

    C < 0.

Output:-

Experiment – 6

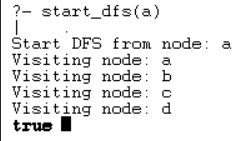
edge(a, b).

edge(b, c).

edge(c, d).

edge(d, a).

edge(b, e).



show\_edges :-

    write('Edges of the graph:'), nl,

    edge(X, Y),

    write(X), write(' -> '), write(Y), nl,

    fail.

show\_edges.

show\_neighbors(Node) :-

    write('Neighbors of '), write(Node), write(':'), nl,

    edge(Node, Neighbor),

    write(Neighbor), nl,

    fail.

show\_neighbors(\_).

show\_graph :-

    show\_edges, nl,

    show\_neighbors(a),

    show\_neighbors(b),

    show\_neighbors(c),

    show\_neighbors(d).

dfs(Node, Visited) :-

    write('Visiting node: '), write(Node), nl,

    edge(Node, Neighbor),

    not(member(Neighbor, Visited)),

    dfs(Neighbor, [Node | Visited]).

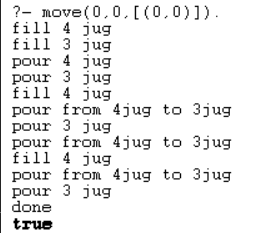
dfs(\_, \_).

start\_dfs(Node) :-

    write('Start DFS from node: '), write(Node), nl,

    dfs(Node, []).

Experiment – 7

Output:-

Code:-

member(X,[X|\_]).

member(X,[Y|Z]):-member(X,Z).

move(X,Y,\_):-X=:=2,Y=:=0,write('done'),!.

move(X,Y,Z):-X<4,\+member((4,Y),Z),write("fill 4 jug"),nl,move(4,Y,[(4,Y)|Z]).

move(X,Y,Z):-Y<3,\+member((X,3),Z),write("fill 3 jug"),nl,move(X,3,[(X,3)|z]).

move(X,Y,Z):-X>0,\+member((0,Y),Z),write("pour 4 jug"),nl,move(0,Y,[(0,Y)|Z]).

move(X,Y,Z):-Y>0,\+member((X,0),Z),write("pour 3 jug"),nl,move(X,0,[(X,0)|Z]).

move(X,Y,Z):-P is X+Y,P>=4,Y>0,K is 4-X,M is Y-K,\+member((4,M),Z),write("pour from 3jug to 4jug"),nl,move(4,M,[(4,M)|Z]).

move(X,Y,Z):-P is X+Y,P>=3,X>0,K is 3-Y,M is X-K,\+member((M,3),Z),write("pour from 4jug to 3jug"),nl,move(M,3,[(M,3)|Z]).

move(X,Y,Z):-K is X+Y,K<4,Y>0,\+member((K,0),Z),write("pour from 3jug to 4jug"),nl,move(K,0,[(K,0)|Z]).

move(X,Y,Z):-K is X+Y,K<3,X>0,\+member((0,K),Z),write("pour from 4jug to 3jug"),nl,move(0,K,[(0,K)|Z]).

Experiment – 8 Code:-

test(Plan):-

    write('Initial state:'),nl,

    Init= [at(tile4,1), at(tile3,2), at(tile8,3),

at(empty,4), at(tile2,5), at(tile6,6), at(tile5,7), at(tile1,8), at(tile7,9)],

    write\_sol(Init),

    Goal= [at(tile1,1), at(tile2,2), at(tile3,3),

at(tile4,4), at(empty,5), at(tile5,6), at(tile6,7), at(tile7,8), at(tile8,9)],

    nl,write('Goal state:'),nl,

    write(Goal),nl,nl,

    solve(Init,Goal,Plan).

solve(State, Goal, Plan):-

    solve(State, Goal, [], Plan).

is\_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 - Y1).

solve(State, Goal, Plan, Plan):-

    is\_subset(Goal, State), nl,

    write\_sol(Plan).

solve(State, Goal, Sofar, Plan):-

    act(Action, Preconditions, Delete, Add),

    is\_subset(Preconditions, State),

    \+ member(Action, Sofar),

    delete\_list(Delete, State, Remainder),

    append(Add, Remainder, NewState),

    solve(NewState, Goal, [Action|Sofar], Plan).

act(move(X,Y,Z),

    [at(X,Y), at(empty,Z), is\_movable(Y,Z)],

    [at(X,Y), at(empty,Z)],

    [at(X,Z), at(empty,Y)]).

is\_subset([H|T], Set):-

    member(H, Set),

    is\_subset(T, Set).

is\_subset([], \_).

delete\_list([H|T], Curstate, Newstate):-

    remove(H, Curstate, Remainder),

    delete\_list(T, Remainder, Newstate).

delete\_list([], Curstate, Curstate).

remove(X, [X|T], T).

remove(X, [H|T], [H|R]):-

    remove(X, T, R).

write\_sol([]).

write\_sol([H|T]):-

    write\_sol(T),

    write(H), nl.

append([H|T], L1, [H|L2]):-

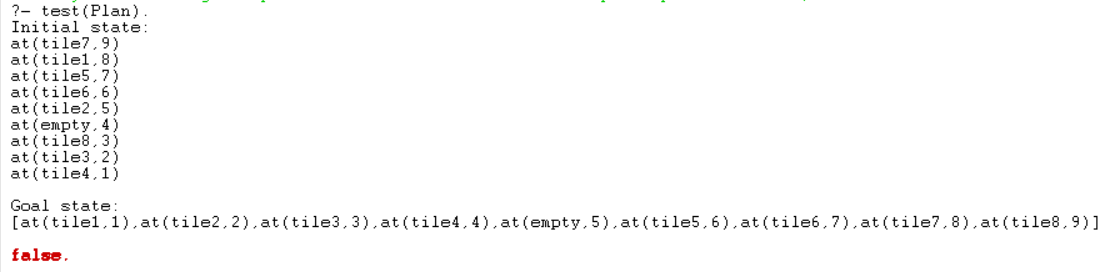
    append(T, L1, L2).

append([], L, L).

member(X, [X|\_]).

member(X, [\_|T]):-

    member(X, T).

Output:-

Experiment – 9

Aim:- Write a program to implement a Tic-Tac-Toe game using Prolog.

Code:-

win(b, p) :- rowwin(b, p).

win(b, p) :- colwin(b, p).

win(b, p) :- diagwin(b, p).

rowwin(b, p) :- b = [p,p,p,\_,\_,\_,\_,\_,\_].

rowwin(b, p) :- b = [\_,\_,\_,p,p,p,\_,\_,\_].

rowwin(b, p) :- b = [\_,\_,\_,\_,\_,\_,p,p,p].

colwin(b, p) :- b = [p,\_,\_,p,\_,\_,p,\_,\_].

colwin(b, p) :- b = [\_,p,\_,\_,p,\_,\_,p,\_].

colwin(b, p) :- b = [\_,\_,p,\_,\_,p,\_,\_,p].

diagwin(b, p) :- b = [p,\_,\_,\_,p,\_,\_,\_,p].

diagwin(b, p) :- b = [\_,\_,p,\_,p,\_,p,\_,\_].

other(x,o).

other(o,x).

game(b, p) :- win(b, p), !, write([p, p, wins]).

game(b, p) :-

  other(p,Otherp),

  move(b,p,Newb),

  !,

  display(Newb),

  game(Newb,Otherp).

move([b,B,C,D,E,F,G,H,I], p, [p,B,C,D,E,F,G,H,I]).

move([A,b,C,D,E,F,G,H,I], p, [A,p,C,D,E,F,G,H,I]).

move([A,B,b,D,E,F,G,H,I], p, [A,B,p,D,E,F,G,H,I]).

move([A,B,C,b,E,F,G,H,I], p, [A,B,C,p,E,F,G,H,I]).

move([A,B,C,D,b,F,G,H,I], p, [A,B,C,D,p,F,G,H,I]).

move([A,B,C,D,E,b,G,H,I], p, [A,B,C,D,E,p,G,H,I]).

move([A,B,C,D,E,F,b,H,I], p, [A,B,C,D,E,F,p,H,I]).

move([A,B,C,D,E,F,G,b,I], p, [A,B,C,D,E,F,G,p,I]).

move([A,B,C,D,E,F,G,H,b], p, [A,B,C,D,E,F,G,H,p]).

display([A,B,C,D,E,F,G,H,I]) :- write([A,B,C]),nl,write([D,E,F]),nl,

 write([G,H,I]),nl,nl.

selfgame :- game([b,b,b,b,b,b,b,b,b],x).

x\_can\_win\_in\_one(b) :- move(b, x, Newb), win(Newb, x).

orespond(b,Newb) :-

  move(b, o, Newb),

  win(Newb, o),

  !.

orespond(b,Newb) :-

  move(b, o, Newb),

  not(x\_can\_win\_in\_one(Newb)).

orespond(b,Newb) :-

  move(b, o, Newb).

orespond(b,Newb) :-

  not(member(b,b)),

  !,

  write('Cats game!'), nl,

  Newb = b.

xmove([b,B,C,D,E,F,G,H,I], 1, [x,B,C,D,E,F,G,H,I]).

xmove([A,b,C,D,E,F,G,H,I], 2, [A,x,C,D,E,F,G,H,I]).

xmove([A,B,b,D,E,F,G,H,I], 3, [A,B,x,D,E,F,G,H,I]).

xmove([A,B,C,b,E,F,G,H,I], 4, [A,B,C,x,E,F,G,H,I]).

xmove([A,B,C,D,b,F,G,H,I], 5, [A,B,C,D,x,F,G,H,I]).

xmove([A,B,C,D,E,b,G,H,I], 6, [A,B,C,D,E,x,G,H,I]).

xmove([A,B,C,D,E,F,b,H,I], 7, [A,B,C,D,E,F,x,H,I]).

xmove([A,B,C,D,E,F,G,b,I], 8, [A,B,C,D,E,F,G,x,I]).

xmove([A,B,C,D,E,F,G,H,b], 9, [A,B,C,D,E,F,G,H,x]).

xmove(b, \_, b) :- write('Illegal move.'), nl.

playo :- explain, playfrom([b,b,b,b,b,b,b,b,b]).

explain :-

  write('You play X by entering integer positions followed by a period.'),

  nl,

  display([1,2,3,4,5,6,7,8,9]).

playfrom(b) :- win(b, x), write('You win!').

playfrom(b) :- win(b, o), write('I win!').

playfrom(b) :- read(N),

  xmove(b, N, Newb),

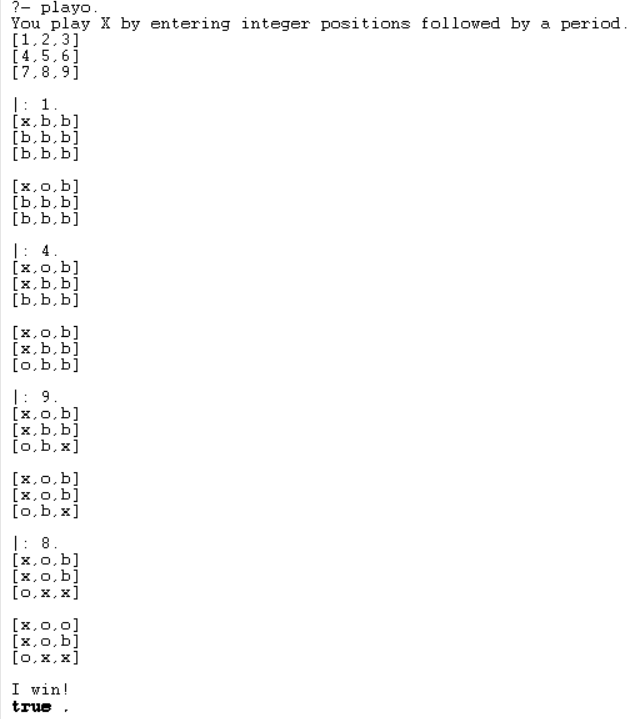
  display(Newb),

  orespond(Newb, Newnewb),

  display(Newnewb),

  playfrom(Newnewb).

Output:-



Experiment – 10

Aim:- Write a program to implement the Hangman game using Python.

Code:-

import random as r

d=r.choice

def h():

w=d(["python","java","kotlin","js","hangman","dev","code","YPS"]);g=set();a=6

print("Hangman!")

while a:

print(" ".join(l if l in g else "\_" for l in w))

i=input("Guess:").lower()

if len(i)!=1 or not i.isalpha()or i in g:continue

g.add(i);a-=(i not in w)

if all(l in g for l in w):print("Win!",w);return

print("Lost!",w)

h()

output:

Hangman!

\_ \_ \_ \_ \_ \_

Guess:k

\_ \_ \_ \_ \_ \_

Guess:h

\_ \_ \_ h \_ \_

Guess:p

p \_ \_ h \_ \_

Guess:y

p y \_ h \_ \_

Guess:t

p y t h \_ \_

Guess:o

p y t h o \_

Guess:n

Win! python

Experiment – 11

Code:-

import nltk

from nltk.stem import PorterStemmer as P

from nltk.tokenize import word\_tokenize as w

nltk.download('punkt')

nltk.download('punkt\_tab')

s=input("Enter: ")

print(" ".join(P().stem(i) for i in w(s)))

output:

[nltk\_data] Downloading package punkt to /home/abhay/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package punkt\_tab to /home/abhay/nltk\_data...

[nltk\_data] Package punkt\_tab is already up-to-date!

Enter: running

run

Experiment – 12

Code:-

import nltk

from nltk.tokenize import word\_tokenize as w

from nltk import pos\_tag as p

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

print(p(w(input("Enter: "))))

output:

[nltk\_data] Downloading package punkt to /home/abhay/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package averaged\_perceptron\_tagger to

[nltk\_data] /home/abhay/nltk\_data...

[nltk\_data] Package averaged\_perceptron\_tagger is already up-to-

[nltk\_data] date!

Enter: I am going to movie.

[('I', 'PRP'), ('am', 'VBP'), ('going', 'VBG'), ('to', 'TO'), ('movie', 'NN'), ('.', '.')]

Experiment – 13

Code:-

import nltk

from nltk.stem import WordNetLemmatizer as L

from nltk.tokenize import word\_tokenize as w

from nltk.corpus import wordnet as wn

nltk.download('punkt')

nltk.download('omw-1.4')

l=L()

def f(wd):return l.lemmatize(wd,pos=wn.VERB)

print(" ".join(f(i) for i in w(input("Enter: "))))

Output:

[nltk\_data] Downloading package punkt to /home/abhay/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package omw-1.4 to /home/abhay/nltk\_data...

[nltk\_data] Package omw-1.4 is already up-to-date!

Enter: running

run

Experiment – 14

Code:-

import nltk

from nltk.classify import NaiveBayesClassifier

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

def fs(sen):

return {word: True for word in word\_tokenize(sen)}

td = []

with open("td.txt", "r") as f:

for line in f:

parts = line.strip().rsplit(" ", 1) # Split only at the last space

if len(parts) == 2 and parts[1] in ["positive", "negative"]:

td.append((parts[0], parts[1]))

train\_set = [(fs(text), label) for text, label in td]

classifier = NaiveBayesClassifier.train(train\_set)

sen = input("Enter a sen: ")

print("Sentiment:", classifier.classify(fs(sen)))

Output:-

[nltk\_data] Downloading package punkt to /home/abhay/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

Enter a sentence: this movie is so great!!!

Sentiment: positive