SSMDA LAB

[DA-304P]

Submitted to: Mr Akshay Mool

Submitted by: Sahil Mishra

Roll No.: 351148027222

Semester: 6th Semester

Group: AIML 2-B

Maharaja Agrasen Institute of Technology, PSP Area,

Sector-22, Rohini, New Delhi-110085

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

M A I T

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware - Software Symbiosis: Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life - Long Learning: The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization: The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification: The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitalization of Learning Processes: The Institute provides seamless opportunities or innovative learning in all Engineering and Management disciplines through digitalization or learning processes using analysis, synthesis, simuation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship: The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they emerge as successful entrepreneurs and responsible citizens.

PRACTICAL RECORD

PAPER CODE : CIE-374P

Name of the student: Lakshya Kumar

University Roll No.: 20714802721

Branch : CSE Shift-I

Section/ Group : FSD – 1C

PRACTICAL DETAILS:

Sno.	*Experiment name*	*Date of performing*	*Date of checking*	*R1 (3)*	*R1 (3)*	*R1 (3)*	*R1 (3)*	*R1 (3)*	*Total Marks (15)*	*Signature*

Sno.	*Experiment name*	*Date of performing*	*Date of checking*	*R1 (3)*	*R2 (3)*	*R3 (3)*	*R4 (3)*	*R5 (3)*	*Total Marks (15)*	*Signature*

Sno.	*Experiment name*	*Date of performing*	*Date of checking*	*R1 (3)*	*R2 (3)*	*R3 (3)*	*R4 (3)*	*R5 (3)*	*Total Marks (15)*	*Signature*

EXPERIMENT -

STUDY OF PROLOG

PROLOG as the name itself suggests, is the short form of **Log**ical **Pro**gramming. It is a logical and declarative programming language.

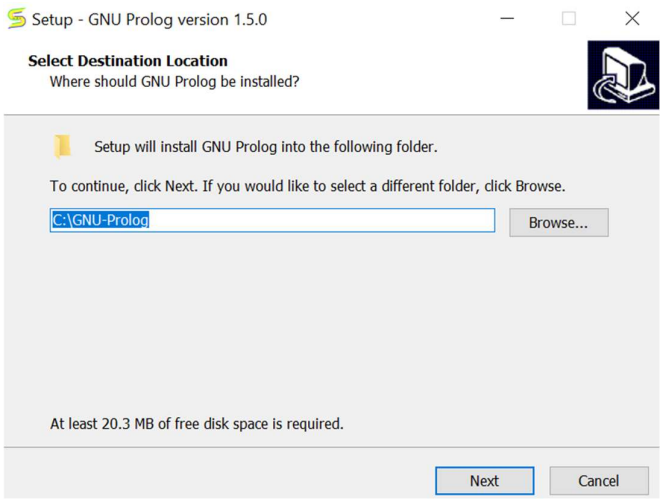
Logic Programming is one of the Computer Programming Paradigm, in which the program statements express the facts and rules about different problems within a system of formal logic.

GNU Prolog is a compiler developed by Daniel Diaz. It is interactive debugging environment for Prolog available for Unix, Windows, MacOS and Linux. It also supports some extensions to Prolog including constraint programming over a finite domain, parsing using definite clause grammars, and an operating system interface.

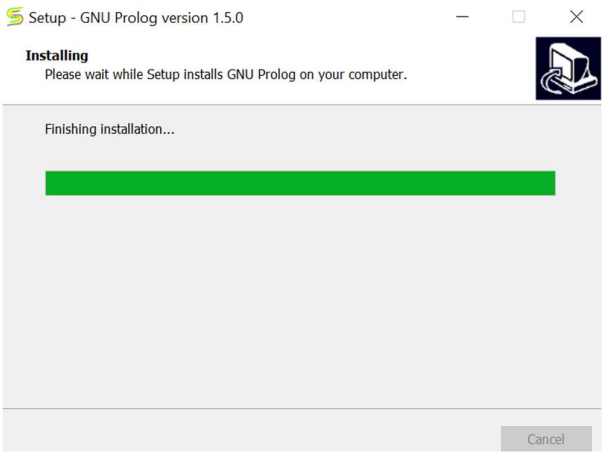
How to download GNU PROLOG?

Install the latest version GNU PROLOG setup file from google.

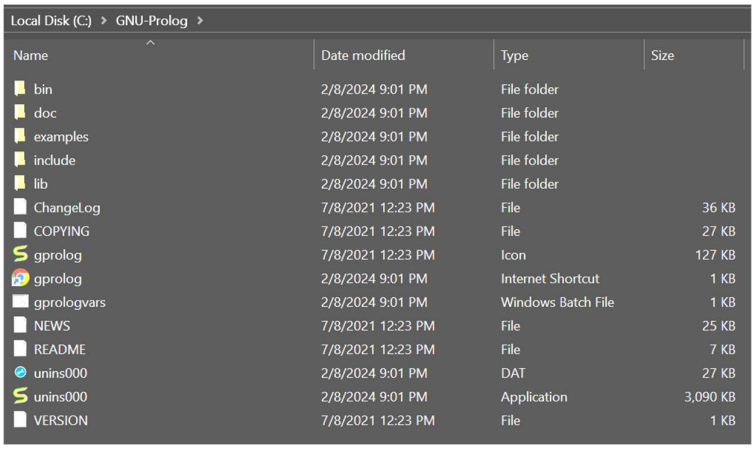
After clicking on the setup file this dialog will open. Select the drive in which you want to install GNU PROLOG.



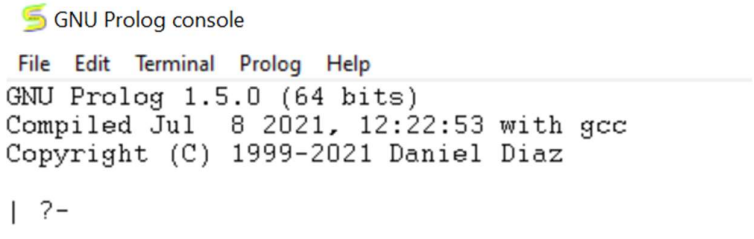
Accept all terms and conditions and it will automatically install the program and the files related to it.



You will see these files inside the GNU PROLOG folder. The experiments will pe performed inside the bin folder.



This is what how the application (GNU PROLOG) will look like when no file is opened.



EXPERIMENT –

Aim:

Write simple fact for the statements using PROLOG

a. Ram likes mango.
b. Seema is a girl.
c. Bill likes Cindy.
d. Rose is red.
e. John owns gold.

Facts:

```
likes(ram,mango).
girl(seema).
likes(bill,cindy).
red(rose).
owns(john,gold).
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/EXP2.pl for byte code...
C:/GNU-Prolog/bin/EXP2.pl:3: warning: discontiguous predicate likes/2 - clause ignored
C:/GNU-Prolog/bin/EXP2.pl compiled, 4 lines read - 625 bytes written, 8 ms
| ?- likes(ram,mango).

yes
| ?- red(rose).

yes
| ?- |
```

EXPERIMENT –

Aim:

Write a program to add two numbers in Prolog.

Source Code:

```
add(X,Y,Result):-Result is X + Y.
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/EXP3.pl for byte code...
C:/GNU-Prolog/bin/EXP3.pl compiled, 0 lines read - 361 bytes written, 13 ms
| ?- add(4,5,Result).

Result = 9

yes
```

EXPERIMENT –

Aim:

Write a program to find factorial of a number in Prolog.

Source Code:

```
factorial(0,1).
factorial(N,M) :-
N>0,
N1 is N-1,
factorial(N1, M1),
M is N*M1.
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/EXP4.pl for byte code...
C:/GNU-Prolog/bin/EXP4.pl compiled, 5 lines read - 824 bytes written, 1 ms
| ?- factorial(5,W).

W = 120 ?
```

EXPERIMENT –

Aim:

Write predicates, one converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing using PROLOG.

Source Code:

```
c_to_f(C,F) :- F is (C * 9 / 5 + 32).
% here freezing point is less than 32 Fahrenheit
freezing(F) :- F =< 32.
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/EXP5.pl for byte code...
C:/GNU-Prolog/bin/EXP5.pl compiled, 3 lines read - 658 bytes written, 4 ms
| ?- c_to_f(20,F).

F = 68.0

yes
```

EXPERIMENT –

Aim:

Write a program to show backtracking in Prolog.

Source Code:

```
boy(tom).
boy(bob).
girl(alice).
girl(lili).

pay(X,Y) :- boy(X), girl(Y).
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/EXP7.pl for byte code...
C:/GNU-Prolog/bin/EXP7.pl compiled, 5 lines read - 677 bytes written, 7 ms
| ?- pay(X,Y).

X = tom
Y = alice ? ;

X = tom
Y = lili ? ;

X = bob
Y = alice ? ;

X = bob
Y = lili

yes
```

EXPERIMENT –

Aim:

Write a program to implement Breath First Search Traversal.

Source Code:

```
% Define edges in the graph
edge(a, b).
edge(a, c).
edge(b, d).
edge(b, e).
edge(c, f).
edge(c, g).

% bfs(Start, Goal, Path) is true if Path is a path from Start to Goal
bfs(Start, Goal, Path) :- bfs_path(Start, Goal, [Start], Path).

% Base case: Goal is the current node
bfs_path(Node, Node, Visited, [Node|Visited]).

% Recursive case: Explore neighbors of the current node
bfs_path(Current, Goal, Visited, Path) :-
   edge(Current, Neighbor),
   \+ member(Neighbor, Visited),
   bfs_path(Neighbor, Goal, [Neighbor|Visited], Path).
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/exp8.pl for byte code...
C:/GNU-Prolog/bin/exp8.pl compiled, 19 lines read - 1583 bytes written, 19 ms
| ?- bfs(a, d, Path).

Path = [d,d,b,a] ?

yes
| ?- bfs(a, f, Path).

Path = [f,f,c,a] ?

yes
| ?-
```

EXPERIMENT –

Aim:

Write a program to implement Water Jug Problem.

Source Code:

```
 % Action rules for pouring water between jugs

% Pour from jug 1 to jug 2
pour(jug1, jug2, [jug(Amount1, Amount2) | Rest], [jug(NewAmount1, NewAmount2) |
Rest]) :-
    Amount1 > 0,
    Amount2 < 3,
    AmountSum is Amount1 + Amount2,
    NewAmount2 is min(AmountSum, 3),
    NewAmount1 is Amount1 - (NewAmount2 - Amount2).

% Pour from jug 2 to jug 1
pour(jug2, jug1, [jug(Amount1, Amount2) | Rest], [jug(NewAmount1, NewAmount2) |
Rest]) :-
    Amount2 > 0,
    Amount1 < 4,
    AmountSum is Amount1 + Amount2,
    NewAmount1 is min(AmountSum, 4),
    NewAmount2 is Amount2 - (NewAmount1 - Amount1).

% Fill jug 1
fill(jug1, [jug(_, Amount2) | Rest], [jug(4, Amount2) | Rest]).

% Fill jug 2
fill(jug2, [jug(Amount1, _) | Rest], [jug(Amount1, 3) | Rest]).

% Empty jug 1
empty(jug1, [jug(_, Amount2) | Rest], [jug(0, Amount2) | Rest]).

% Empty jug 2
empty(jug2, [jug(Amount1, _) | Rest], [jug(Amount1, 0) | Rest]).

% Check if the target amount is reached
target_reached([jug(_, 2) | _]).

% Depth-first search to find a solution
dfs(Start, _, Visited, Actions) :-
    target_reached(Start),
    reverse(Visited, Actions).
```

```
dfs(State, DepthLimit, Visited, Actions) :-
    DepthLimit > 0,
    DepthLimit1 is DepthLimit - 1,
    (pour(_, _, State, NextState);
     fill(_, State, NextState);
     empty(_, State, NextState)),
    \+ member(NextState, Visited),
    dfs(NextState, DepthLimit1, [NextState | Visited], Actions).

% Predicate to find a solution
find_solution(Start, MaxDepth, Actions) :-
    dfs(Start, MaxDepth, [Start], Actions).
```

PROLOG:

Actions =
[[jug(0,0)],[jug(4,0),jug(0,0)],[jug(1,3),jug(4,0),jug(0,0)],[jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(0,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(3,0),jug(0,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(3,3),jug(3,0),jug(0,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)],[jug(4,2),jug(3,3),jug(3,0),jug(0,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(1,3),jug(4,0),jug(0,0)]] ?

EXPERIMENT –

Aim:

Write a program to remove punctuations from the given string.

Source Code:

```
% Define a predicate to remove punctuations from a string
remove_punctuations(Input, Output) :-
    atom_chars(Input, InputChars),
    filter_punctuations(InputChars, CleanChars),
    atom_chars(Output, CleanChars).

% Define a predicate to filter out punctuation characters
filter_punctuations([], []).
filter_punctuations([Char|Rest], CleanChars) :-
    (punctuation_char(Char) -> filter_punctuations(Rest, CleanChars) ; CleanChars =
[Char|CleanRest], filter_punctuations(Rest, CleanRest)).

% Define a predicate to check if a character is a punctuation character
punctuation_char(Char) :-
    member(Char, ['.', ',', ';', ':', '!', '?', '"', '\'', '-', '(', ')']).
```

PROLOG:

```
C:/GNU-Prolog/bin/exp10.pl compiled, 14 lines read - 2387 bytes written, 9 ms

(16 ms) yes
| ?- remove_punctuations('Hello, world!', Output).

Output = 'Hello world'

yes
```

EXPERIMENT –

Aim:

Write a program to sort the sentence in alphabetical order.

Source Code:

```
 % Base case: If the list is empty, it is already sorted
sort_sentence([], []).

% Sort the sentence by finding the minimum word in the list
sort_sentence(Sentence, [Min|Sorted]) :-
    find_min(Sentence, Min),
    remove_from_list(Sentence, Min, Remaining),
    sort_sentence(Remaining, Sorted).

% Find the minimum word in the list
find_min([Word], Word). % Base case: The minimum of one element is itself
find_min([Word1,Word2|Rest], Min) :-
    Word1 @=< Word2, % If Word1 is less than or equal to Word2
    find_min([Word1|Rest], Min).
find_min([Word1,Word2|Rest], Min) :-
    Word1 @> Word2, % If Word1 is greater than Word2
    find_min([Word2|Rest], Min).

% Remove an element from the list
remove_from_list([], _, []).
remove_from_list([X|Xs], X, Xs).
remove_from_list([Y|Ys], X, [Y|Zs]) :-
    remove_from_list(Ys, X, Zs).
```

PROLOG:

```
compiling C:/GNU-Prolog/bin/exp11.pl for byte code...
C:/GNU-Prolog/bin/exp11.pl compiled, 23 lines read - 2264 bytes written, 11 ms
| ?- sort_sentence([hello, world, this, is, a, test], SortedSentence).

SortedSentence = [a,hello,is,test,this,world] ? |
```

EXPERIMENT –

Aim:

Write a program to implement Tic-Tac-Toe game.

Source Code:

```
 win(Board, Player) :- rowwin(Board, Player).
win(Board, Player) :- colwin(Board, Player).
win(Board, Player) :- diagwin(Board, Player).

rowwin(Board, Player) :- Board = [Player,Player,Player,_,_,_,_,_,_].
rowwin(Board, Player) :- Board = [_,_,_,Player,Player,Player,_,_,_].
rowwin(Board, Player) :- Board = [_,_,_,_,_,_,Player,Player,Player].

colwin(Board, Player) :- Board = [Player,_,_,Player,_,_,Player,_,_].
colwin(Board, Player) :- Board = [_,Player,_,_,Player,_,_,Player,_].
colwin(Board, Player) :- Board = [_,_,Player,_,_,Player,_,_,Player].

diagwin(Board, Player) :- Board = [Player,_,_,_,Player,_,_,_,Player].
diagwin(Board, Player) :- Board = [_,_,Player,_,Player,_,Player,_,_].
% Helping predicate for alternating play in a "self" game:

other(x,o).
other(o,x).

game(Board, Player) :- win(Board, Player), !, write([player, Player, wins]).
game(Board, Player) :-
  other(Player,Otherplayer),
  move(Board,Player,Newboard),
  !,
  display_board(Newboard),
  game(Newboard,Otherplayer).

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

display_board([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).

% Predicates to support playing a game with the user:

x_can_win_in_one(Board) :- move(Board, x, Newboard), win(Newboard, x).

% The predicate orespond generates the computer's (playing o) reponse
% from the current Board.

orespond(Board,Newboard) :-
  move(Board, o, Newboard),
  win(Newboard, o),
  !.
orespond(Board,Newboard) :-
  move(Board, o, Newboard),
  not(x_can_win_in_one(Newboard)).
orespond(Board,Newboard) :-
  move(Board, o, Newboard).
orespond(Board,Newboard) :-
  not(member(b,Board)),
  !,
  write('Cats game!'), nl,
  Newboard = Board.

% The following translates from an integer description
% of x's move to a board transformation.

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(Board, _, Board) :- write('Illegal move.'), nl.

% The 0-place predicate playo starts a game with the user.

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_board([1,2,3,4,5,6,7,8,9]).
```

```
playfrom(Board) :- win(Board, x), write('You win!').
playfrom(Board) :- win(Board, o), write('I win!').
playfrom(Board) :- read(N),
  xmove(Board, N, Newboard),
  display_board(Newboard),
  orespond(Newboard, Newnewboard),
  display_board(Newnewboard),
  playfrom(Newnewboard).
```

PROLOG:

```
yes
| ?- selfgame.
[x,b,b]
[b,b,b]
[b,b,b]

[x,o,b]
[b,b,b]
[b,b,b]

[x,o,x]
[b,b,b]
[b,b,b]

[x,o,x]
[o,b,b]
[b,b,b]

[x,o,x]
[o,x,b]
[b,b,b]

[x,o,x]
[o,x,o]
[b,b,b]

[x,o,x]
[o,x,o]
[x,b,b]

[x,o,x]
[o,x,o]
[x,o,b]

[player,x,wins]
```

EXPERIMENT – 12

Aim:

Write a program to implement Hangman game using python.

Source Code:

```
import random

# List of words to choose from
words = ['apple', 'banana', 'orange', 'grape', 'kiwi', 'pear']

def choose_word(words):
    """Choose a random word from the list."""
    return random.choice(words)

def display_word(word, guessed_letters):
    """Display the word with underscores for unguessed letters."""
    display = ''
    for letter in word:
        if letter in guessed_letters:
            display += letter + ' '
        else:
            display += '_ '
    return display.strip()

def hangman():
    """Main function to play the Hangman game."""
    # Choose a word
    word = choose_word(words)
    # Initialize variables
    guessed_letters = []
    attempts = 6

    print("Welcome to Hangman!")
    print("Try to guess the word.")
    print(display_word(word, guessed_letters))

    # Main game loop
    while True:
        guess = input("Enter a letter: ").lower()

        if guess in guessed_letters:
            print("You already guessed that letter.")
            continue
        elif len(guess) != 1 or not guess.isalpha():
```

```
            print("Please enter a single letter.")
            continue

        guessed_letters.append(guess)

        if guess not in word:
            attempts -= 1
            print("Incorrect guess! Attempts remaining:", attempts)
            if attempts == 0:
                print("You ran out of attempts! The word was:", word)
                break
        else:
            print("Correct guess!")

        # Display current state of the word
        display = display_word(word, guessed_letters)
        print(display)

        # Check if the word has been completely guessed
        if '_' not in display:
            print("Congratulations! You guessed the word:", word)
            break

# Play the game
hangman()
```

OUTPUT:

```
Welcome to Hangman!
Try to guess the word.

_ _ _ _ _ _
Enter a letter: e
Incorrect guess! Attempts remaining: 5

_ _ _ _ _ _
Enter a letter: t
Incorrect guess! Attempts remaining: 4

_ _ _ _ _ _
Enter a letter: a
Correct guess!
_ a _ a _ a
Enter a letter: m
Incorrect guess! Attempts remaining: 3
_ a _ a _ a
Enter a letter: l
Incorrect guess! Attempts remaining: 2
_ a _ a _ a
Enter a letter: r
Incorrect guess! Attempts remaining: 1
_ a _ a _ a
Enter a letter: v
Incorrect guess! Attempts remaining: 0
You ran out of attempts! The word was: banana
```

EXPERIMENT – 13

Aim:

Write a program to implement Hangman game.

Source Code:

```
    create_guess_list(Length, GuessList),
    play(WordList, GuessList).

% Helper predicate to create a list of underscores.
create_guess_list(Length, GuessList) :-
    length(GuessList, Length),
    maplist(=(' _ '), GuessList).

% Predicate to handle the game logic.
play(WordList, GuessList) :-
    write('Current word: '), write(GuessList), nl,
    write('Enter a letter: '), read(Letter),
    (   member(Letter, WordList)
    ->  update_guess_list(WordList, GuessList, Letter, NewGuessList),
        (   WordList == NewGuessList
        ->  write('Congratulations! You have guessed the word: '), write(WordList), nl
        ;   play(WordList, NewGuessList)
        )
    ;   write('Sorry, that letter is not in the word.'), nl,
        play(WordList, GuessList)
    ).

% Predicate to update the guessed letters list.
update_guess_list([], [], _, []).
update_guess_list([H|T], [G|GT], Letter, [H|NT]) :-
    (   H == Letter
    ->  update_guess_list(T, GT, Letter, NT);
    update_guess_list(T, GT, Letter, [G|NT])
    ).
update_guess_list([H|T], [' _ '|GT], Letter, [' _ '|NT]) :-
    update_guess_list(T, GT, Letter, NT).

% Example of starting the game with the word 'prolog'.
% ?- hangman('prolog').
```

PROLOG:

```
?- hangman('microsoft').
Current word: [ _ , _ , _ , _ , _ , _ , _ , _ , _ ]
Enter a letter: 'r'.
Current word: [m,c,o,o, _ ]
Enter a letter: |: 'm'.
Current word: [m,c,o, _ ,f]
Enter a letter: |: 'o'.
Current word: [m,c,o, _ , _ , _ ]
Enter a letter: |: 'f'.
Current word: [m,c, _ ,s,f]
Enter a letter: |: 's'.
Current word: [m,c, _ ,s, _ , _ ]
Enter a letter: |: 'i'.
Current word: [m,c, _ , _ ,o, _ ]
Enter a letter: |:
```

EXPERIMENT – 14

Aim:

Write a program to remove stop words for a given passage from a text file using NLTK.

Source Code:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('stopwords')
nltk.download('punkt')

def remove_stopwords(text):
    tokens = word_tokenize(text)

    stop_words = set(stopwords.words('english'))

    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

    filtered_text = ' '.join(filtered_tokens)

    return filtered_text

def main():
    # Modify the path, use raw string or double backslashes
    file_path = r'D:\College Labs\SEM 6\AI\test.txt'

    try:
        with open(file_path, 'r') as file:
            passage = file.read()

        cleaned_passage = remove_stopwords(passage)

        print("Original Passage:\n", passage)
        print("\nCleaned Passage:\n", cleaned_passage)
    except FileNotFoundError:
        print("File not found at the specified path.")
    except Exception as e:
        print("An error occurred:", e)

main()
```

OUTPUT:

Original Passage:
Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur est accusantium reprehenderit aspernatur? Aliquam ipsum consequuntur voluptatem corrupti veritatis ratione mollitia laborum atque quae nulla delectus praesentium facilis qui, provident nemo quis libero asperiores eos veniam dignissimos cupiditate ipsa. Rem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Repellendus, eaque exercitationem. Porro at incidunt repellat corrupti nisi? Ea non saepe officiis ab, rem necessitatibus aliquid culpa voluptatum, reprehenderit, explicabo molestias. Sequi ducimus quibusdam laudantium tenetur dolor aut nulla. Quis odit explicabo, corrupti, doloremque eum eius asperiores voluptas expedita, recusandae excepturi alias molestiae ullam nostrum? Possimus temporibus sit saepe cupiditate cumque delectus, perspiciatis inventore fuga dolor alias, maxime unde quaerat beatae porro, ex laboriosam explicabo aperiam illo. Animi nobis sed rerum voluptatibus et incidunt dolorem expedita obcaecati, vel nisi ab tempore omnis quas quibusdam illum minus suscipit possimus, placeat quidem iure, sunt nam pariatur hic labore! Explicabo officia corrupti consectetur voluptatibus dolor incidunt quia dolore dolores quaerat eos. Corrupti, rem animi.

Cleaned Passage:
Lorem ipsum dolor sit amet consectetur adipisicing elit . Consectetur est accusantium reprehenderit aspernatur ? Aliquam ipsum consequuntur voluptatem corrupti veritatis ration
e mollitia laborum atque quae nulla delectus praesentium facilis qui , provident nemo quis libero asperiores eos veniam dignissimos cupiditate ipsa . Rem . Lorem ipsum dolor , s
it amet consectetur adipisicing elit . Repellendus , eaque exercitationem . Porro incidunt repellat corrupti nisi ? Ea non saepe officiis ab , rem necessitatibus aliquid culpa v
oluptatum , reprehenderit , explicabo molestias . Sequi ducimus quibusdam laudantium tenetur dolor aut nulla . Quis odit explicabo , corrupti , doloremque eum eius asperiores vo
luptas expedita , recusandae excepturi alias molestiae ullam nostrum ? Possimus temporibus sit saepe cupiditate cumque delectus , perspiciatis inventore fuga dolor alias , maxim
e unde quaerat beatae porro , ex laboriosam explicabo aperiam illo . Animi nobis sed rerum voluptatibus et incidunt dolorem expedita obcaecati , vel nisi ab tempore omnis quas q
uibusdam illum minus suscipit possimus , placeat quidem iure , sunt nam pariatur hic labore ! Explicabo officia corrupti consectetur voluptatibus dolor incidunt quia dolore dolo
res quaerat eos . Corrupti , rem animi .

EXPERIMENT – 15

Aim:

Write a program to implement stemming for a given sentence using NLTK.

Source Code:

```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

nltk.download('punkt')

def stem_sentence(sentence):
    tokens = word_tokenize(sentence)

    porter = PorterStemmer()

    stemmed_tokens = [porter.stem(word) for word in tokens]

    stemmed_sentence = ' '.join(stemmed_tokens)

    return stemmed_sentence

def main():
    # Input sentence
    sentence = "Natural language processing is a field of artificial intelligence
that focuses on the interaction between computers and humans through natural
language."

    stemmed_sentence = stem_sentence(sentence)

    print("Original Sentence:\n", sentence)
    print("\nStemmed Sentence:\n", stemmed_sentence)

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Original Sentence:
 Natural language processing is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language.

Stemmed Sentence:
 natur languag process is a field of artifici intellig that focus on the interact between comput and human through natur languag .
```

EXPERIMENT – 16

Aim:

Write a program to POS (part of speech) tagging for the give sentence using NLTK.

Source Code:

```
 import nltk

from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

def pos_tag_sentence(sentence):
    # Tokenize the sentence
    tokens = word_tokenize(sentence)

    # Perform POS tagging
    tagged_tokens = pos_tag(tokens)

    return tagged_tokens

def main():
    # Input sentence
    sentence = "Natural language processing is a field of artificial intelligence
that focuses on the interaction between computers and humans through natural
language."

    # Perform POS tagging on the sentence
    tagged_sentence = pos_tag_sentence(sentence)

    # Print the original and tagged sentences
    print("Original Sentence:\n", sentence)
    print("\nTagged Sentence:\n", tagged_sentence)

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Original Sentence:
 Natural language processing is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language.

Tagged Sentence:
 [('Natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('field', 'NN'), ('of', 'IN'), ('artificial', 'JJ'), ('intelligence', 'NN'), ('that',
'WDT'), ('focuses', 'VBZ'), ('on', 'IN'), ('the', 'DT'), ('interaction', 'NN'), ('between', 'IN'), ('computers', 'NNS'), ('and', 'CC'), ('humans', 'NNS'), ('through', 'IN'), ('n
atural', 'JJ'), ('language', 'NN'), ('.', '.')]
```

EXPERIMENT – 17

Aim:

Write a program to implement Lemmatization using NLTK.

Source Code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')

def lemmatize_sentence(sentence):
    tokens = word_tokenize(sentence)

    lemmatizer = WordNetLemmatizer()

    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in tokens]

    lemmatized_sentence = ' '.join(lemmatized_tokens)

    return lemmatized_sentence

def main():
    sentence = "Natural language processing is a field of artificial intelligence
that focuses on the interaction between computers and humans through natural
language."

    lemmatized_sentence = lemmatize_sentence(sentence)

    print("Original Sentence:\n", sentence)
    print("\nLemmatized Sentence:\n", lemmatized_sentence)


main()
```

OUTPUT:

```
Original Sentence:
 Natural language processing is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language.

Lemmatized Sentence:
 Natural language processing is a field of artificial intelligence that focus on the interaction between computer and human through natural language .
```

EXPERIMENT – 18

Aim:

Write a program for Text Classification for the given sentence using NLTK.

Source Code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.classify import NaiveBayesClassifier
import random

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

def preprocess(sentence):
    # Tokenize the sentence
    tokens = word_tokenize(sentence.lower())

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize tokens
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]

    return lemmatized_tokens

def extract_features(words):
    return dict([(word, True) for word in words])

def train_classifier():
    # Sample training data
    training_data = [
        (preprocess("Natural language processing is a field of artificial
intelligence."), "technology"),
        (preprocess("Forests are home to diverse ecosystems."), "nature"),
        (preprocess("Computers can understand and generate human language."),
"technology"),
        (preprocess("Mountains offer breathtaking views and fresh air."), "nature"),
```

```
        (preprocess("Machine learning algorithms improve with more data."),
"technology"),
        (preprocess("Rivers provide water for plants and animals."), "nature")
    ]

    # Extract features from training data
    training_features = [(extract_features(tokens), category) for tokens, category
in training_data]

    # Train Naive Bayes classifier
    classifier = NaiveBayesClassifier.train(training_features)

    return classifier

def classify_sentence(classifier, sentence):
    tokens = preprocess(sentence)

    features = extract_features(tokens)

    category = classifier.classify(features)

    return category

def main():
    classifier = train_classifier()

    test_sentences = [
        "The internet has revolutionized communication.",
        "Birds migrate to warmer climates during winter.",
        "Artificial intelligence is shaping the future of technology.",
        "Forests play a crucial role in maintaining ecological balance.",
        "Deep learning models require large datasets for training.",
        "Oceans cover more than 70% of the Earth's surface.",
        "Blockchain technology is transforming various industries.",
        "Wildlife conservation is essential for biodiversity."
    ]

    # Classify test sentences
    for sentence in test_sentences:
        category = classify_sentence(classifier, sentence)
        print(f"Sentence: {sentence}")
        print(f"Category: {category}")
        print("-" * 50)

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Sentence: The internet has revolutionized communication.
Category: technology
--------------------------------------------------
Sentence: Birds migrate to warmer climates during winter.
Category: technology
--------------------------------------------------
Sentence: Artificial intelligence is shaping the future of technology.
Category: technology
--------------------------------------------------
Sentence: Forests play a crucial role in maintaining ecological balance.
Category: nature
--------------------------------------------------
Sentence: Deep learning models require large datasets for training.
Category: technology
--------------------------------------------------
Sentence: Oceans cover more than 70% of the Earth's surface.
Category: technology
--------------------------------------------------
Sentence: Blockchain technology is transforming various industries.
Category: technology
--------------------------------------------------
Sentence: Wildlife conservation is essential for biodiversity.
Category: technology
--------------------------------------------------
```