TABLE OF CONTENTS

S. No	Date	Aim of the Experiment	Signature/date
1.	28/1/22	Study of Prolog	
2.	7/1/22	a) WAP in Prolog to print all the elements of the list. b) WAP in Prolog to find whether an element is the member of the list or not. c)WAP in Prolog to extract the Kth element from the list. d) Write a program in Prolog to implement sumlist (List,Sum) so that Sum is the sum of a given list of numbers List. e) Write a Prolog program to find whether the first number is greater than, less than	
3.	14/2/22	a) Write a Prolog program to implement countlist (List,Count) so that Count is the count of given list of numbers List. b) Write a program in prolog to concatenate one list to the other list.	
4.	14/2/22	a) Write a program in Prolog to find the factorial of a number. b) Write a program in Prolog to show the working of the following built-in predicates of Prolog	
5.	21/3/22	a) Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively. b) Write a Prolog program to perform append, delete and replace using lists. c) Write a program in prolog to count up from a number to 10.	

6.	5/4/22	Write a program in prolog to implement Best First Search Algorithm.	
7.	21/3/22	a) Write a program in prolog to create the state space for the following and check the connections that exist between any two nodes. b) Write a program in prolog to find the path between any two states of the following state space. c) Write a program in prolog to find the path between any two states of the following state space. d) Write a program in prolog to accept the input from the user and perform string matching.	
8.	5/4/22	Write a program in prolog to implement Depth First Search Algorithm.	
9.	19/4/22	Write a program in Python to Implement K-means clustering algorithm.	
10.	19/4/22	Write a program in Python to solve any one AI problem.	

EXPERIMENT NO - 10 **DATE: 19/4/22**

AIM OF THE EXPERIMENT: WRITE A PROGRAM IN PYTHON TO SOLVE ANY ONE AI PROBLEM

SOURCE CODE :

Below is the code of a Deep Learning General Purpose Chat Bot, using the LSTM model.

```
""LSTM_ChatBot.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1sSuaffc-
DfB3nNneRwjFHaOQ2GH9sr1W
"""

import tensorflow as tf
import numpy as np
import pandas as pd
import json
import nltk
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Input, Embedding, LSTM ,
Dense,GlobalMaxPooling1D,Flatten
from tensorflow.keras.models import Model
import matplotlib.pyplot as plt

#importing the dataset
with open('/content/database.json') as content:
  data1 = json.load(content)
#getting all the data to lists
tags = []
inputs = []
responses={}
for intent in data1['intents']:
  responses[intent['tag']]=intent['responses']
  for lines in intent['patterns']:
    inputs.append(lines)
    tags.append(intent['tag'])
#converting to dataframe
data = pd.DataFrame({"inputs":inputs,
                     "tags":tags})
```

```
print(data)

#removing punctuations
import string
data['inputs'] = data['inputs'].apply(lambda wrd:[ltrs.lower() for
ltrs in wrd if ltrs not in string.punctuation])
data['inputs'] = data['inputs'].apply(lambda wrd: ''.join(wrd))
#tokenize the data
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=2000)
tokenizer.fit_on_texts(data['inputs'])
train = tokenizer.texts_to_sequences(data['inputs'])

#apply padding
from tensorflow.keras.preprocessing.sequence import pad_sequences
x_train = pad_sequences(train)

#encoding the outputs
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(data['tags'])

#input length
input_shape = x_train.shape[1]
print(input_shape)
#define vocabulary
vocabulary = len(tokenizer.word_index)
print("number of unique words : ",vocabulary)
#output length
output_length = le.classes_.shape[0]
print("output length: ",output_length)

#creating the model
i = Input(shape=(input_shape,))
x = Embedding(vocabulary+1,10)(i)
x = LSTM(10,return_sequences=True)(x)
x = Flatten()(x)
x = Dense(output_length,activation="softmax")(x)
model  = Model(i,x)
#compiling the model
model.compile(loss="sparse_categorical_crossentropy",optimizer='adam
',metrics=['accuracy'])
#training the model
```

```
train = model.fit(x_train,y_train,epochs=500)

#chatting
import random
while True:
  texts_p = []
  prediction_input = input('You : ')
  #removing punctuation and converting to lowercase
  prediction_input = [letters.lower() for letters in
prediction_input if letters not in string.punctuation]
  prediction_input = ''.join(prediction_input)
  texts_p.append(prediction_input)
  #tokenizing and padding
  prediction_input = tokenizer.texts_to_sequences(texts_p)
  prediction_input = np.array(prediction_input).reshape(-1)
  prediction_input = pad_sequences([prediction_input],input_shape)
  #getting output from model
  output = model.predict(prediction_input)
  output = output.argmax()
  #finding the right tag and predicting
  response_tag = le.inverse_transform([output])[0]
  print("Alice: ",random.choice(responses[response_tag]))
  if response_tag == "goodbye":
    break
```

OUTPUT:

```
You : hey
Alice:  Hello
You : who are you
Alice:  Alice
You : make me laugh
Alice:  I own the world's worst thesaurus. Not only is it awful, it's awful.
You : bye
Alice:  Have a nice day
```

Fig 10.1 Output