Lab Manual

PROGRAM-1(A)

1A) Aim: Write a program to print the multiplication table for the given number?

Program:

```
# Python program to find the multiplication table (from 1 to 10) of a number input by the
 user # take input from the user

num = int(input("Display multiplication table of? "))

# use for loop to iterate 10 times

 for i in range(1,11):
        print(num,'x',i,'=',num*i)
```

Output:

```
Display multiplication table of? 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

1B) Aim: Write a program to find factorial of the given number?

Program:

```
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)
num = int(input("Enter a number: "))
# check is the number is negative
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    print("The factorial of",num,"is",recur_factorial(num))
```

Program 3:

Artificial Intelligence Lab A.Y: 2023-24

PROGRAM-3

3) **Aim:** Write a program to implement simple Chat bot?

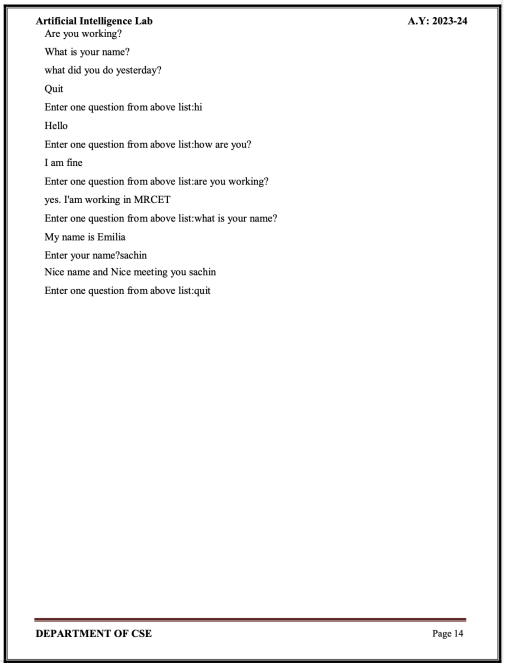
Program:

```
print("Simple Question and Answering Program")
print("=========================================")
print(" You may ask any one of these questions")
print("Hi")
print("How are you?")
print("Are you working?")
print("What is your name?")
print("what did you do yesterday?")
print("Quit")
while True:
    question = input("Enter one question from above list:")
    question = question.lower()
    if question in ['hi']:
        print("Hello")
    elif question in ['how are you?','how do you do?']:
        print("I am fine")
    elif question in ['are you working?','are you doing any job?']:
        print("yes. I'am working in MRCET")
    elif question in ['what is your name?']:
        print("My name is Emilia")
        name=input("Enter your name?")
        print("Nice name and Nice meeting you",name)
    elif question in ['what did you do yesterday?']:
        print("I saw Bahubali 5 times")
    elif question in ['quit']:
        break
    else:
        print("I don't understand what you said")
```

Output:

```
Simple Question and Answering Program
=====================================
You may ask any one of these questions
Hi
How are you?
```

DEPARTMENT OF CSE Page 13



Artificial Intelligence Lab A.Y: 2023-24

```
Are you working?
What is your name?
what did you do yesterday?
Quit
Enter one question from above list:hi
Hello
Enter one question from above list:how are you?
I am fine
Enter one question from above list:are you working?
yes. I'am working in MRCET
Enter one question from above list:what is your name?
My name is Emilia
Enter your name?sachin
Nice name and Nice meeting you sachin
Enter one question from above list:quit
```

DEPARTMENT OF CSE Page 14

Program 4:
Machine Learning: Given, you have the music.csv file s below.
Write a program in python to train the data and make prediction for you.

Solution.

```
# importing the panda library and giving it shortcut as pd
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
# import data
music_data=pd.read_csv('music.csv')
# splitting data
X=music_data.drop(columns=['genre'])
y=music_data['genre']
# creating model
model=DecisionTreeClassifier()
# Train the model
model.fit(X,y)
# make predictions
predictions=model.predict([[21,1],[22,0]])
//output the predictions
predictions
```

Python

```
/Users/chiranjiviregmi/anaconda3/lib/python3.11/site-packages/sklearn/base.py:464: UserWarning: X does not have valid feature names, but DecisionTreeC
  warnings.warn(

array(['HipHop', 'Dance'], dtype=object)
```

Note:Please write the css file content as well in the lab manual.

music

age	gender	genre
20	1	HipHop
23	1	HipHop
25	1	HipHop
26	1	Jazz
29	1	Jazz
30	1	Jazz
31	1	Classical
33	1	Classical
37	1	Classical
20	0	Dance
21	0	Dance
25	0	Dance
26	0	Acoustic
27	0	Acoustic
30	0	Acoustic
31	0	Classical
34	0	Classical
35	0	Classical

Prolog:

Prolog:

Aim:

Study of PROLOG Programming language and its Functions. Write simple facts for the statements using PROLOG.

OBJECTIVE: Study of PROLOG.

PROLOG-PROGRAMMING IN LOGIC

PROLOG stands for Programming, In Logic — an idea that emerged in the early 1970's to use logic as programming language. The early developers of this idea included Robert Kowaiski at Edinburgh (on the theoretical side), Marrten van Emden at Edinburgh (experimental demonstration) and Alian Colmerauer at Marseilles (implementation).

David D.H. Warren's efficient implementation at Edinburgh in the mid -1970's greatly contributed to the popularity of PROLOG. PROLOG is a programming language centred around a small set of basic mechanisms, Including pattern matching, tree based data structuring and automatic backtracking. This Small set constitutes a surprisingly powerful and flexible programming framework. PROLOG is especially well suited for problems that involve objects- in particular, structured objects- and relations between them.

SYMBOLIC LANGUAGE

PROLOG is a programming language for symbolic, non-numeric computation. It is especially well suited for solving problems that involve objects and relations between objects.

For example, it is an easy exercise in PROLOG to express spatial relationship between objects, such as the blue sphere is behind the green one. It is also easy to state a more general rule: if object X is closer to the observer than object Y. and object Y is closer than Z, then X must be closer than Z. PROLOG can reason about the spatial relationships and their consistency with respect to the general rule. Features like this make PROLOG a powerful language for ArtJIcia1 LanguageA1,) and non- numerical programming.

There are well-known examples of symbolic computation whose implementation in other standard languages took tens of pages of indigestible code, when the same algorithms were implemented in PROLOG, the result was a crystal-clear program easily fitting on one page.

1.1.1 Knowledge Base 1

Knowledge Base 1 (KB1) is simply a collection of facts. Facts are used to state things that are unconditionally true of the domain of interest. For example, we can state that Mia, Jody, and Yolanda are women, and that Jody plays air guitar, using the following four facts:

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
```

This collection of facts is KB1. It is our first example of a Prolog program. Note that the names `mia`, `jody`, and `yolanda`, and the properties `woman` and `playsAirGuitar`, have been written so that the first letter is in lower-case. This is important; we will see why a little later.

How can we use KB1? By posing queries. That is, by asking questions about the information KB1 contains. Here are some examples. We can ask Prolog whether Mia is a woman by posing the query:

```
?- woman(mia).
```

Prolog will answer

```
yes
```

for the obvious reason that this is one of the facts explicitly recorded in KB1. Incidentally, *we* don't type in the `?-`. This symbol (or something like it, depending on the implementation of Prolog you are using) is the prompt symbol that the Prolog interpreter displays when it is waiting to evaluate a query. We just type in the actual query (for example `woman(mia)`) followed by `.` (a full stop).

Similarly, we can ask whether Jody plays air guitar by posing the following query:

```
?- playsAirGuitar(jody).
```

Prolog will again answer ``yes", because this is one of the facts in KB1. However, suppose we ask whether Mia plays air guitar:

```
?- playsAirGuitar(mia).
```

We will get the answer

```
no
```

Why? Well, first of all, this is not a fact in KB1. Moreover, KB1 is extremely simple, and contains no other information (such as the *rules* we will learn about shortly) which might help Prolog try to *infer* (that is, *deduce* whether Mia plays air guitar. So Prolog correctly concludes that `playsAirGuitar(mia)` does *not* follow from KB1.

Here are two important examples. Suppose we pose the query:

```
?- playsAirGuitar(vincent).
```

Again Prolog answers ``no". Why? Well, this query is about a person (Vincent) that it has no information about, so it concludes that `playsAirGuitar(vincent)` cannot be deduced from the information in KB1.

Similarly, suppose we pose the query:

```
?- tatooed(jody).
```

Again Prolog will answer ``no". Why? Well, this query is about a property (being tatooed) that it has no information about, so once again it concludes that the query cannot be deduced from the information in KB1.

FACTS, RULES AND QUERIES

Programming in PROIOG is accomplished by creating a database of facts and rules about objects, their properties, and their relationships to other objects. Queries then can be posed about the objects and valid conclusions will be determined and returned by the program Responses to user queries are determined through a form of inference control known as resolution.

FOR EXAMPLE:

a. Ram likes mango.

b. Seema is a girl.

c. Bill likes Cindy.

d. Rose is red.

e. John owns gold.

Program:

Clauses

likes(ram ,mango).

girl(seema).

red(rose).

likes(bill ,cindy).

owns(john ,gold).

Output:

queries

?-likes(ram,What).

What= mango

?-likes(Who,cindy).

Who= cindy

?-red(What).

What= rose

?-owns(Who,What).

Who= john

What= gold.

Program to demonstrate a simple prolog program.

predicates

like(symbol,symbol)
hate(symbol,symbol)

clauses

like(sita,ram).
like(x,y).
like(a,b).

hate(c,d).
hate(m,n).

hate(f,g).

Output:-