

INDEX

NAME Dhanya Vanisha A. (220401062) SUBJECT _____
 STD. _____ DIV. _____ ROLL NO. _____ SCHOOL _____

SR. NO.	PAGE NO. <u>Date</u>	TITLE	DATE marks	TEACHER'S SIGN / REMARKS
1.		Python Programs	9	Top
2.		Project : Details.	9	Top
3.		Code for DFS Search.	9	Top
4.		N-Queens Problem	10	Top
5.		A* Algorithm	10	Top
6.		Water Jug Problem.	10	Top
7.		Implementation of decision tree classification techniques	10	Top
8.		Implementation of K-Means clustering technique.	10	Top
9.		Implementation of Artificial Neural Networks for an Application in Regression	10	Top
10.		Introduction to Prolog -	10	Top
11.		Prolog family tree	10	Top
12.		Minimax Algorithm.	10	Top

Completed

Top

```

1. def factorial(n):
2     if n > 0:
3         return n * factorial(n-1)
4     elif n == 0 or n == 1:
5         return 1
6     else:
7         fact = 1
8         while (n > 1):
9             fact *= n
10            n -= 1
11        return fact
12
13 num = int(input("Enter a number: "))
14 print("Factorial of", num, "is", factorial(num))

```

Output : enter a number : 4
 Factorial of 4 is 24

```

2. def largest(arr, n):
3     max = arr[0]
4     for i in range(1, n):
5         if arr[i] > max:
6             max = arr[i]
7     return max
8
9 if __name__ == '__main__':
10    arr = [10, 324, 45, 90, 9808]
11    n = len(arr)
12    ans = largest(arr, n)
13    print("Largest in given array is", ans)

```

Output : Largest in given array is 9808

3) def palindrome(s):

 return s == s[:: -1]

s = "malayalam"

ans = palindrome(s)

if ans:

 print("Yes")

else

 print("No")

Output: Yes

(a) bivalent (b)

o in 11

a in 10

b in 01

c in 00

d in 11

(c) stable

n = 100

4) def armstrong(n):

 sum = 0

 temp = n

 while temp > 0: (a) bivalent (b) b1 = num

 digit = temp % 10 (c) bivalent (d) b2 = num

 sum += digit ** 3

 temp = temp // 10 (e) bivalent (f) b3 = num

 if n == sum: (g) bivalent (h) b4 = num

 print(n, "is a armstrong number")

 else

 print(n, "is not an armstrong number")

n = 153

armstrong(n)

(i) bivalent (j) b5 = num

num < 1000 (k) b6 = num

l7mo = num

output: 153 is an armstrong number.

5) def prime(a, b): (l) b7 = num

 for num in range(a, b + 1):

 if num > 1: (m) b8 = num

 for ref in range(2, num):

 if (num % i) == 0:

 break (n) b9 = num

 else:

 print(num)

a = 1

b = 10

prime (a, b)

Output: 11 21 27 31 37 41 47

3

5

7

turnout affected?

existing pipeline system at least 10 km's distance away

new pipeline: 2 hectare biomass production at pipeline 10 km's distance

biofuel plant 2 hectare land with biomass 20 ha's area produced

biofuel plant for more biomass production to explore market

with about 20% probability for biomass with probability

about 10% biomass of agricultural market for biofuel

about 10% less cost compared to energy market

with about one tenth monthly a two dozen of oil imports

switching biomass market to about 20% oil imports due

production increase demand biomass from suppliers

biofuels at higher bio share inhibited no biofuels

Chances reduce time

20. difficulties involved

a robot for detection biomass harvesting left the

environmental detection of sensor may change to remove

the biomass leaves or dust between sensor of sensor this

with 10% less bio fuel solution because more smoke

more smoke will give more smoke, smoke to remove

more biomass no bio fuel solution to remove, smoke

a smidge less biomass solution because of

smoke removal from sensor, a lot of smoke will remove

dust removal solution this is a problem solution

30. write briefly about - what off timing a bio

harvesting biomass their growth may not occur maximum

maximum turn biomass two days of crop biomass

plant, and older leaves out in the

dominate for biomass production which is not possible

35. what are the difficulties involved

ProjectFUTURE INVESTMENT PREDICTION (FIP)

5

Problem Statement

Most investors find it hard to make intelligent decisions due to the difficulty in handling financial markets: cryptocurrencies, stocks, and valuable metals. They lack the tools that aid in easy analysis of past and present data of the market, including the capacity of predicting future trends that help in tailoring investment strategies to maximize gains within a shorter period of time. Hence, the need for this project is to work out a platform that can handle these challenges by its service of data-driven insights, predictive analysis and personalized investment recommendations based on individual goals with respect to financial and market dynamics.

20

Problem Description

In the high-speed financial markets of today, it becomes difficult for investors to make the right decisions with respect to various assets, such as cryptocurrencies, stocks and precious metals like gold and silver. Huge volumes of data, along with the volatile nature of markets, make it really hard for an individual investor to discover profitable opportunities and optimize his investment strategy. Besides, more than ever, timely and accurate predictions along with personalized recommendations are required. The above-discussed platform aims to empower users by processing their investment preferences and financial goals to give out strategies that maximize profit in the shortest possible time, thereby bridging the gap between complicated & actionable insights.

30

35

Abstract:

It propose an all-new investment platform mastering cutting-edge data analytics and machine learning to analyze historical and real-time data across cryptocurrency, stock and precious metal-gold and silver, markets. This investment platform will be capable of integrating time series analysis, predictive modeling, and sentiment analysis in a manner that derives forecasts related to prospective price movements and market trends. Users can input the investment amount, which, coupled with market data, the system processes to come up with relevant personally important investment strategies. This tries to maximize the user profit within a time constraint by using data-driven actionable insights and optimized portfolio allocation across multiple asset classes.

Target audience:

Investors' needs range from those of retail investors, needing basic or sophisticated tools on investing in cryptocurrencies, to enthusiasts looking for more data-driven insights. Precious metals and collectibles investors zero in on market predictions and intrinsic value. The risk-averse, including retirees, look forward to safety, which DIY investors would like the proper tools that will help them in making decision. HNWIs pursue tailored solutions, while educational institutions implement learning platforms. Technology and AI lovers discover another frontier, innovative finance solutions.

Objective:

The objective is to create a platform that provides an accurate price prediction and personalized investment recommendation system in line with user inputs, all targeted towards securing the best possible investment strategy to be adopted in order to yield maximum profitability within a short period. Integrate data-driven insight with predictive modeling, in a manner that

allows users to make informed decisions towards better financial outcomes over the long term.

5. Financial Services: Financial services include banking, insurance, and investment management. Our fintech project focuses on leveraging innovative financial technologies and updated analytics to predict market trends and optimize investments. Key areas

10. involving include applying machine learning, deep learning, AI, and NLP for price prediction and sentiment analysis, using quantitative finance for asset pricing and strategy development, and investing in software development to ensure robust user interfaces and data integration.

~~15. Not only businesses but also individuals can benefit from AI-powered solutions. Individuals can use AI-powered tools to manage their personal finances, track expenses, and plan for retirement.~~

15

16. Healthcare: Healthcare is another sector that has been transformed by AI. In medicine, AI is used for diagnostic imaging, such as X-rays and CT scans, to detect diseases early. It is also used for drug discovery, where AI algorithms analyze large amounts of data to identify potential new drugs. Another application of AI in healthcare is in the form of robotic surgery, which allows doctors to perform complex operations with greater precision and accuracy. Overall, AI has revolutionized the healthcare industry, making it more efficient and accessible to people around the world.

30

31. Manufacturing: Manufacturing is another industry that has been transformed by AI. In manufacturing, AI is used for quality control, where AI algorithms analyze data from sensors to detect defects in products. It is also used for process optimization, where AI algorithms analyze data from various sources to identify inefficiencies in the manufacturing process and suggest ways to improve it. Additionally, AI is used for predictive maintenance, where AI algorithms analyze data from machinery to predict when maintenance will be required, reducing downtime and increasing efficiency. Overall, AI has revolutionized the manufacturing industry, making it more efficient and cost-effective.

35

1. N-Queens Problem

Aim: To write the code for N-Queens Problem.

Code:

```
N = int(input("Enter N:"))
```

```
board = [[0, 0, 0, 0]]
```

```
[[0, 0, 0, 0]]
```

```
def solveNQueens(board, col):
```

```
if col == N:
```

```
    print(board)
```

```
    return True
```

```
for i in range(N):
```

```
    if isSafe(board, i, col):
```

```
        board[i][col] = 1
```

```
        if solveNQueens(board, col + 1):
```

```
            return True
```

```
        board[i][col] = 0
```

```
    return False
```

```
def isSafe(board, row, col):
```

```
    for x in range(col):
```

```
        if board[row][x] == 1:
```

```
            return False
```

```
    for x, y in zip(range(row, -1, -1), range(col, -1, -1)):
```

```
        if board[x][y] == 1:
```

```
            return False
```

```
    for x, y in zip(range(row, N, 1), range(col, 1, -1)):
```

```
        if board[x][y] == 1:
```

```
            return False
```

```
return True
```

```
board = [[0 for x in range(N)] for y in range(N)]
```

```
if not solveNQueens(board, 0):
```

```
    print("No Solution found")
```

Outputfor N=4:

Enter N=4 at 1203 11:45 am with

[[0, 0, 1, 0],

[1, 0, 0, 0],

[0, 0, 0, 1], (1, 0) being 1st row

[0, 1, 0, 0]]

(1, 0, 0, 0) is 1st row

for N=8:

at 1203 11:

(1, 0, 0) being

Enter N=8 at 1203 11:45 am with

[[1, 0, 0, 0, 0, 0, 0, 0], 1],

[0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0],

:([0, 0, 0, 0, 0, 0, 0, 0], 1],

[0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0],

:([0, 0, 0, 0, 0, 0, 0, 1], 1].

Result:

:(1, 0, 0) is 1st row

thus, the program for N-Queens problem have been executed & verified successfully.

:(1, 0, 0) is 1st row, (1, 1, 0, 0) is 2nd row

:(1, 0, 1, 0) is 3rd row,

etc 7 rows!

(1, 0, 0) is 1st row, (1, 0, 0, 0, 0, 0, 0, 0) is 8th row

(0, 1, 0, 0, 0, 0, 0, 0) is 9th row

(0, 0, 1, 0, 0, 0, 0, 0) is 10th row

a) Depth first Search

Aim: To write a program for depth first search algorithm.

Code: + to discuss of writing a class of

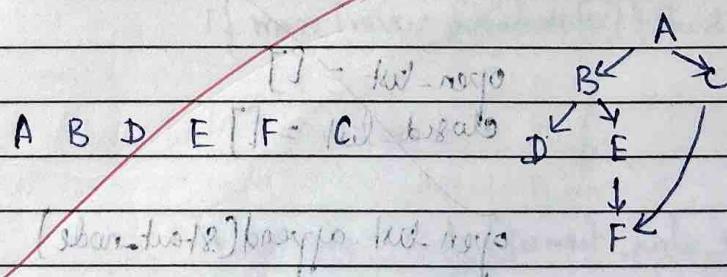
5 def dfs(graph, start, visited=None):
 if visited is None:
 visited = set()
 visited.add(start)
 print(f"start, end = {start, end}")

10 for neighbour in graph[start]:
 if neighbour not in visited:
 dfs(graph, neighbour, visited)

graph = {
 'A': ['B', 'C'],
 'B': ['D', 'E'],
 'C': ['F'],
 'D': [],
 'E': [],
 'F': []}

(dfs(graph), 'A') = abc-def
 ('A', start) = abc - abc

25 Output:



Result: o < (fail-safe)nd visited

This] + the program for depth first search has been executed & verified successfully.

fail-safe) because if not, what if

35 f. base cases & f. not f
 not f = base. break

visit = visit + base

3) A* search algorithm:Bim:

To write a program to execute A* search algorithm.

Code:

```
import heapq
```

```
class Node:
```

```
    def __init__(self, parent=None, position=None):
```

```
        self.parent = parent
```

```
        self.position = position
```

```
        self.g = 0
```

```
        self.h = 0
```

```
        self.f = 0
```

```
def eq(self, other):
```

```
    return self.position == other.position
```

```
def astar(maze, start, end):
```

```
    start_node = Node(None, start)
```

```
    end_node = Node(None, end)
```

25 A

~~open-list = []~~

~~closed-list = []~~

~~open-list.append(start_node)~~

30

```
while len(open-list) > 0 :
```

```
    current_node = open-list[0]
```

```
    current_index = 0
```

```
    for index, item in enumerate(open-list):
```

```
        if item.f < current_node.f:
```

```
            current_node = item
```

```
            current_index = index
```

open-list.pop(index).blue
 closed-list.append(current-node)

if current-node != end-node:

path = []

current = current-node

while current is not None:

path.append(current.position)

current = current.parent

return path[:: -1].reverse()

children = []

for new-position in [(0, +1), (0, -1), (+1, 0), (-1, 0),
 (+1, +1), (+1, -1), (-1, +1), (-1, -1)]:

node-position = current-node.position[0] +
 new-position[0], current-node.position[1] +
 new-position[1]

+ new-position[2]

+ new-position[3]

+ new-position[4]

+ new-position[5]

+ new-position[6]

+ new-position[7]

continue

(0, 0) - true

if maze[node-position[0]][node-position[1]]
 [node-position[2]] == 0:

(base, true, continue) - stop

else (true)

new-node = Node(current-node, node-position)

children.append(new-node)

for child in children:

for closed-child in closed-list:

if child == closed-child:

continue.

child.g = current-node.g + 1 - weight

```
child.b = ((child.position[0]-end_node.position[0])*++2)
          + ((child.position[1]-end_node.position[1])*++1)
```

$$\text{child-f} \rightarrow \text{child-g} + \text{child-h}$$

for open-node in open-list:

$\text{child} == \text{open_node}$ and $\text{child.g} > \text{open_key}$

~~18-19. Continue~~ 18-19. ~~Continue~~

open wt. apprend. (child.)

```
def main();
```

U - subfix

$$\text{maze} = [[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],$$

$$\therefore (-1, -1, 1, 1), ([0, 0, 0, 0, 1, 0, 0, 0, 0, 0],$$

15 ~~asid 189. abn. locn. [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]~~,
16 ~~asid 190. abn. locn. [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]~~

$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$[0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0]$,
 $[0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0]$

$$= ((1 - 0.01)^{10})^{10} = 0.99^{10} \approx 0.904$$

$$\mathbf{g}_{\text{start}} = (0, 0)$$

inf₂₅ of what) / [ə] end = (y, 1.6) at 2011

~~path = astar(maze, start, end)~~

print (path)

~~eg über das herum geht - das war~~

name: _____

most likely to have had

Output is block size of

~~Ex:~~ $\{ (0,0), (1,1), (2,2), (3,3), (4,4), (5,5), (6,6) \}$

Result: \rightarrow exit node

Thus, the python program for A* algorithm is verified & executed successfully.

4) Water jug problem:

Aim:

To write a program to execute Water Jug problem.

program code:

```

def fill_4-gallon(x,y,x-max,y-max):
    return (x-max,y)

def fill_3-gallon(x,y,x-max,y-max):
    return (x,y-max)

def empty_4-gallon(x,y,x-max,y-max):
    return (0,y)

def empty_3-gallon(x,y,x-max,y-max):
    return (x,0)

def pour_4-to_3(x,y,x-max,y-max):
    transfer = min(x,y-max-y)
    return (x-transfer,y+transfer)

def pour_3-to_4(x,y,x-max,y-max):
    transfer = min(y,x-max-x)
    return (x+transfer,y-transfer)

def off-water-jug(x-max,y-max,goal-x,
                  visited=None, start=(0,0)):

    if visited is None:
        visited = set()

    stack = [start]
    while stack:
        state = stack.pop()
        x,y = state
        if state in visited:
            continue
        visited.add(state)
        print(f"Visiting state: {state}")
        if x == goal-x:
            print(f"Goal reached: {state}")
            return state

```

next-states = [fill-1-gallon($x, y, x\text{-max}, y\text{-max}$),
 fill-3-gallon($x, y, x\text{-max}, y\text{-max}$),
 empty-1-gallon($x, y, x\text{-max}, y\text{-max}$),
 empty-3-gallon($x, y, x\text{-max}, y\text{-max}$),
 pour-2-to-3($x, y, x\text{-max}, y\text{-max}$),
 pour-3-to-2($x, y, x\text{-max}, y\text{-max}$)]

5

10

for new-state in next-states :

if new-state not in visited :

stack.append(new-state)

else return none.

15

$x\text{-max} = 4$

$y\text{-max} = 3$

goal-x = 2

20

dfs-water-jug($x\text{-max}, y\text{-max}$, goal-x)

Output:

Visiting state : (0, 0)

Visiting state : (0, 3)

Visiting state : (3, 0)

Visiting state : (3, 3)

Visiting state : (4, 2)

Visiting state : (4, 0)

Visiting state : (1, 3)

Visiting state : (1, 0)

Visiting state : (0, 1)

Visiting State = (4, 1)

Visiting State : (2, 3)

Visiting State : (2, 0)

Goal reached : (2, 3)

(2, 3)

30

nodes [nodo] and nodes [no de] for node and node

Result:

thus, the python ~~executes~~ program to solve water jug problem is executed and verified successfully

for

5. Implementation of decision tree classification techniques

Aim:

5 To implement a decision tree classification technique for gender classification using python.

Explanation:

- ↳ Import tree from sklearn
- ↳ call the fn. Decision Tree Classifier()
- ↳ from tree
- ↳ Assign values for x and y
- ↳ call the function predict for predicting on the basis of given random values for each given feature
- ↳ Display the output-

Program code:

```
20 from sklearn.tree import DecisionTreeClassifier
x = [[170, 65, 40], [160, 55, 38], [180, 80, 42],
      [195, 75, 41], [158, 52, 34]]
```

$$y = [1, 0, 1, 1, 0]$$

Clf = clf.fit(x, y)

prediction = clf.predict([[165, 60, 39]])

print("Predicted Gender: ", "Male", if prediction[0]
 == 1 else "female")

Output:

Predicted Gender: Female.

~~Result:~~

Thus the python program to implement a decision tree classification technique for gender classification is executed successfully

6. Implementation of K-Means clustering technique.

Aim:

To implement a K-Means clustering technique using python language.

Explanation:

- ↳ Import kMeans from sklearn.cluster
- ↳ Assign X and Y
- ↳ Call the function kMeans()
- ↳ Perform scatter operation and display the output.

Program code:

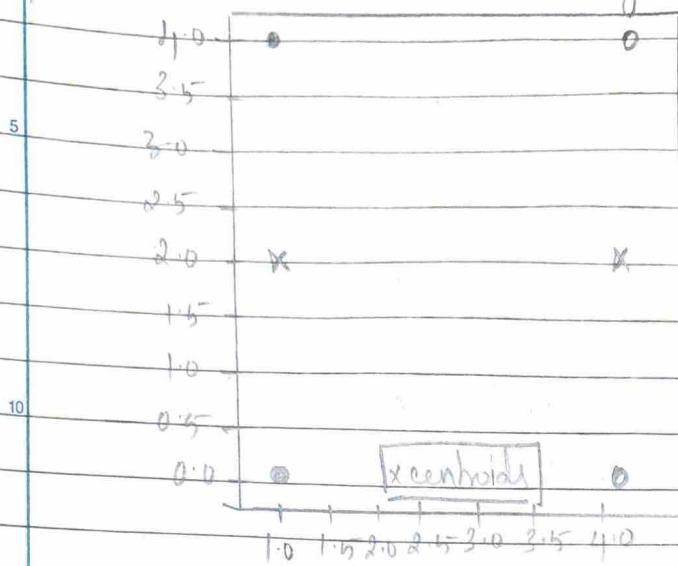
```

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
X = [[1, 2], [1, 4], [1, 0], [2, 2], [2, 4], [4, 0]]
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
Y = kmeans.labels_
centers = kmeans.cluster_centers_
for i, label in enumerate(Y):
    plt.scatter(X[i][0], X[i][1], color='blue' if label == 0 else 'green')
plt.scatter(centers[:, 0], centers[:, 1], color='red', marker='x', s=100, label='centroids')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('K-Means Clustering')
plt.legend()
plt.show()

```

Output : Screenshot of Jupyter Notebook

k-Means Clustering



Result:

thus, the python program to implement
k-means clustering technique is executed
successfully.

4. Implementation of Artificial Neural Networks for An Application in Regression

Aim:

To implement artificial neural networks for an application in regression using python.

Program code:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
X = np.array([[1200, 3, 20], [1500, 4, 15], [1800, 3, 30],
[2000, 5, 10], [1600, 4, 25]])
```

```
y = np.array([20000, 25000, 24000, 30000, 24000])
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
model = Sequential()
```

```
model.add(Dense(32, input_dim=X.shape[1],
activation='relu'))
```

```
model.add(Dense(1))
```

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
model.fit(X_train, Y_train, epochs=10, batch_size=5, verbose=1)
```

```
predictions = model.predict(X_test)
```

```
for i in range(min(15, len(predictions))):  
    print(f"Predicted: {predictions[i][0]}; Actual: {y_tst[i]}")
```

5 Output :

Predicted : 0.06, Actual = 25000.0

10

15

20

25

Result:

Thus, the python program to implement artificial neural networks for a application in regression is executed successfully.

30

35

8. Introduction to Prolog

Aim:

To learn PROLOG terminologies and write basic programs

Terminologies:

1. Atomic terms:-

Atomic terms are usually things made up of lower- and uppercase letters, digits and the underscore, starting with a lowercase letter.

e.g.: dog

ab-c-321

2. Variables:

Variables are string of letters, digits and the underscore, starting with a capital letter or an underscore.

e.g.: Dog

apple-420

3. Compound terms:-

Compound terms are made up of PROLOG atom and a number of arguments (PROLOG terms, i.e., atoms, numbers, variables, or other compound terms) enclosed in parenthesis and separated by commas.

(e.g.:-

is-bigg(x, elephant, X)

f(x, 1), 4)

4. Facts:-

A fact \Leftrightarrow a predicate followed by a dot.

Ex. elephant(john). :- (elephant) known & valid

(is) bigg(animal(whale)). (calvinidiot)

(elephant) known & is - beautiful. (is) known & valid

5. Rules:-

A rule consists of a head (a predicate) and a body (a sequence of predicates separated by commas).

Ex. :- (is_smaller(X, Y)) :- is_bigg(Y, X)

aunt(Aunt, Child) :- Sister(Aunt, Parent), parent(Parent, Child)

Source code:

KB1 :

woman(mia).

woman(judy).

woman(cyolanda).

playsAirGuitar(mia).

party.

query 1: ? - woman(mia).

query 2: ? - playsAirGuitar(mia).

query 3: ? - party.

query 4: ? - concert.

Output - Siphi, called 4 queries as follows.

true.

? - playsAirGuitar(mia).

false

base to query ? - party, there are several answers

redress mode : true. and of course there are several answers to return a

query such as ? - concert and perhaps 1000 or, whatever

Error: Unknown procedure: concert/o (DWIM could not convert
goal)

KB2:

happy(cyolanda).

listens2music(mia).

Listen2music(cyolanda) :- happy(cyolanda).

playsAirGuitar(mia); listens2music(mia)

playsAirGuitar(cyolanda); - listens2music(cyolanda).

so has (classifications) back to source due to

playAirGuitar(mia); - playsAirGuitar(mia).

true

(x,y) ? - playsAirGuitar(cyolanda).

so has (classifications) back to source due to

(x,y), true.

KB3:

likes(dan, Sally)

likes(Sally, dan)

likes(john, brittney)

married(X, Y) :- likes(X, Y), likes(Y, X).

friends(X, Y) :- likes(X, Y); likes(Y, X).

• (lwm) 100, 100) true

• ((pred) 100, 100) true

• ((inv) 100, 100) false

• ((pred) 100, 100) true

• (lwm) 100, 100) false

Output:

? - likes(dan, X).

X = Sally

? - married(dan, Sally).

true.

? - married(john, brittney).

false

• (X, not) true - ?

• (pred) 100 - X

• (lwm) 100 - ?

• (not) = not

• (X) not, (X, not) true - ?

KB4:

food(burgo).

food(sandwich).

food(pizza).

lunch(sandwich).

dinner(pizza). smartest meal of the daymeal(X) :- food(X). meals are categorizedOutput:

? - food(pizza).

true

? - meal(X), lunch(X).

X = sandwich

? - dinner(sandwich).

false

KB5:

owns (jack, car(bmw)).

owns (john, car(chery)).

owns (olivia, car(eivic)).

owns (jane, car(chery)).

sedan (car(bmw)).

sedan (car(eivic)).

truck (car(chery)).

5

Output:

? - owns (john, x).

x = car(chery).

? - owns (john, x).

true.

15

? - owns (who, car(eivic)).

who = john.

? - own (jane, x), sedan(x).

false

? - own (jane, x), truck(x).

x = car(chery)

20

~~Result:~~

Thus, the basic programs consisting of PROLOG terminologies are successfully built and executed.

25

30

35

9. Prolog family tree

Aim:

To develop a family tree program using PROLOG with all possible facts, rules and queries.

knowledge Base:

/* facts

male (peter)

male (john)

male (chris)

male (kelvin)

female (betty)

female (jenny)

female (lisa)

female (helen)

parent of (chris, peter)

parent of (chris, betty)

parent of (helen, peter)

parent of (helen, betty)

parent of (kelvin, chris)

parent of (kelvin, lisa)

parent of (jenny, john)

parent of (jenny, kelvin)

/* Rules

a) father (x, y) :- male (y), parent of (x, y)

Y Y

peter

chris

father

peter

Helen

john

jenny

chris

kelvin

K85:

owns(jack, car(bmw)).

owns(john, car(chery)).

owns(olivia, car(civic)).

owns(jane, car(chery)).

Sedan(car(bmw)).

Sedan(car(civic)).

Truck(car(chery)).

true.

Output:

?- owns(john, x). . (x, nob) with - p

x = car(chery). nob - x

?- owns(john, x). . (x, nob) between - p

true. end

?- owns(who, car(chery)). nob

Who = john. end

?- own(jane, x), sedan(x).

false

?- owns(jane, x), truck(x). (x, nob) both

x = car(chery) . (x, nob) both

. (assig) both

Result:

Thus, the basic programs consisting of PROLOG terminologies are successfully built and executed.

. (assig) both - p

end

(D)omit, (x)both - p

. (x, nob) both - x

. (x, nob) both - p

sdef

9. Prolog family tree

Aim:

→ to develop a family tree program using PROLOG with all possible facts, rules and queries.

knowledge Base:

1 * fact

male(peter)

male(john)

male(chris)

male(kelvin)

female(betty)

female(jeny)

female(lisa)

female(helen)

parent_of(chris, peter)

parent_of(chris, betty)

parent_of(helen, peter)

parent_of(helen, betty)

parent_of(kelvin, chris)

parent_of(kelvin, lisa)

parent_of(jeny, john)

parent_of(jeny, kelvin)

1 * Rule

a) father(X, Y) :- male(Y), parent_of(X, Y)

X Y

peter

chris

john

jeny

chris

kelvin

b) female(Y), parentOf(X,Y)

Y Y

betty Chir

Helen Helen is parent of a parent of

tisa tisa is parent of a parent of

helen Jenny

c) Male(X), parentOf(X,2), parentOf(Z,Y)

Y X Z (sibling) class

peter kelvin chir (sibling) class

peter jenny Helen (sibling) class

(sibling) class

d) female(Y), parentOf(X,2) & parentOf(Z,Y)

Y X Z (sibling) class

betty kelvin chir (sibling) class

betty Jenny Helen (sibling) class

(sibling) class

e) Male(Y), father(X,2), father(Y,W), Z = W

procedure father(A,B) does not exist

f) female(Y), father(X,2), father(Y,W), Z = W

procedure father(A,B) does not exist

~~(Y,X) father, (Y) class : (Y,X) is not (Z,W)~~

~~Result:~~

thus, the prolog family meeting executed successfully.

10. MINIMAX Algorithm

Atm:

↳ A simple example can be used to explain how the minimax algorithm works we have included in this example.

↳ There are 2 players in this scenario one named Maximize and other named Minimize.

↳ Maximizes will strive for higher possible score's and minimizes will strive for lowest score.

Program:

```

import math
def minimax (depth, node_index, is_maximise,
            scores, height):
    if depth == height:
        return scores[node_index]
    if is_maximise:
        return max (minimax (depth + 1, node_index * 2,
                             False, scores, height),
                    minimax (depth + 1, node_index * 2 + 1, True,
                             scores, height))
    else:
        return min (minimax (depth + 1, node_index * 2,
                             False, scores, height),
                    minimax (depth + 1, node_index * 2 + 1, True,
                             scores, height))

```

~~def calculate_tree_height (num_leaves):~~

~~return math.ceil (math.log2 (num_leaves))~~

~~scores = list (map (int, input ("Enter the score\nseparated by a space(' ')").split ()))~~

~~tree_height = calculate_tree_height (0, 0, True,~~

~~scores, tree_height),~~

~~optimal_score = minimax (0, 0, True, scores, tree_height).~~

Date :-

~~print("The optimal score is: " + str(optimal_score))~~

5. *Leptobasis* used to belong in *Microlophus* xanthostictus
is placed with *Leptobasis*

10. *Microlophus* *leopardinus* *leopardinus* has distinct
black stripes along midline of body & no white
intermediate bands between black stripes

Hall Brosit

35 min x 200-400 x 100-200 (high) minimum 100

Gipiel, 602

Opieh - High 91

Freiburg-Baifeldsches Projekt

descriptions. 8

Digitized by srujanika@gmail.com

20 1: Stylus) received (Dipt. char. 169)

((Edged, was a plot of six acres that

100

Output: $\{A_1, A_2, \dots, A_n\}$ (minimum) size nodes

Enter the words specified by, Space (s)

~~answ: x=17, y=4 x=12 y=6 w = 8 H= 5 x=9 y=7~~

The optimal score is 4. (highest)

~~(lowest min) lowest est. estimate fib~~

~~(Conc. phenol. After) fine glass melted~~

Thus, the minimax algorithm successfully determines the optimal move for players.

~~sent~~(i , o, c) \rightarrow i ist sent, es besteht = i ist ein

(Wiel een), drw?

~~new~~ new (not 0.0) remains: new-tempo

(Boked)