

# **Government Engineering College Gandhinagar**



## **B.E in Computer Engineering Semester-N, Year 2022-23 ARTIFICIAL INTELLIGENCE**

**Name**  
**Niraj Italiya**

**Enrollment Number**  
**190130107041**

**Government Engineering College, Sector  
28, Gandhinagar**

**B.E**

**in**

**Computer Engineering**

**Semester-06 Year 2022-23**

**ARTIFICIAL INTELLIGENCE**

**Lab Faculty  
Coordinator**

**Course : -BE**

# **Institute Vision/Mission**

## **Vision:**

- To be a premier engineering institution, imparting quality education for innovative solutions relevant to society and environment.

## **Mission:**

- To develop human potential to its fullest extent so that intellectual and innovative engineers can emerge in a wide range of professions.
- To advance knowledge and educate students in engineering and other areas of scholarship that will best serve the nation and the world in future.
- To produce quality engineers, entrepreneurs and leaders to meet the present and future needs of society as well as environment.

# **Computer Engineering Department Vision/Mission**

## **Vision:**

- To achieve excellence for providing value based education in Computer Engineering through innovation, team work and ethical practices.

## **Mission:**

- To produce computer science and engineering graduates according to the needs of industry, government, society and scientific community.
- To develop partnership with industries, government agencies and R &

## **D Organizations**

- To motivate students/graduates to be entrepreneurs.
- To motivate students to participate in reputed conferences, workshops, symposiums, seminars and related technical activities.

# **Program Educational Outcome (PEO)**

1. To provide students with a strong foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering

problems and to prepare them for graduate studies, R&D, consultancy and higher learning.

2. To develop an ability to analyze the requirements of the software, understand the technical specifications, design and provide novel engineering solutions and efficient product designs.
3. To provide exposure to emerging cutting edge technologies, adequate training & opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.
4. To prepare the students for a successful career and work with values & social concern bridging the digital divide and meeting the requirements of Indian and multinational companies.
5. To promote student awareness on the life-long learning and to introduce them to professional ethics and codes of professional practice

## Program Specific Outcomes (PSO)

By the completion of Computer Engineering program the student will have following Program specific outcomes.

1. Design ,develop, test and evaluate computer based systems by applying standard software engineering practices and strategies in the area of algorithms, web design, data structure, and computer network
2. Apply knowledge of ethical principles required to work in a team as well as to lead a team

## Program Outcomes (PO)

**Engineering Graduates will be able to:**

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological changes.

**INDEX**

<b>Sr.no</b>	<b>Name of Practicals</b>	<b>Date</b>	<b>Sign</b>
1	Write a program to implement Tic-Tac-Toe game problem.		
2	Write a program to implement BFS (for 8 puzzle problem or Water Jug problem or any AI search problem)		
3	Write a program to implement DFS (for 8 puzzle problem or Water Jug problem or any AI search problem)		
4	Write a program to implement Single Player Game (Using any Heuristic Function)		
5	Write a program to Implement A* Algorithm.		
6	Write a program to implement mini-max algorithm for any game development.		
7	Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2).		
8	<p>Define a predicate brother(X,Y) which holds iff X and Y are brothers.</p> <p>Define a predicate cousin(X,Y) which holds iff X and Y are cousins.</p> <p>Define a predicate grandson(X,Y) which holds iff X is a grandson of Y.</p> <p>Define a predicate descendent(X,Y) which holds iff X is a descendent of Y.</p> <p>Consider the following genealogical tree:  father(a,b). father(a,c). father(b,d). father(b,e).  father(c,f).</p> <p>Say which answers, and in which order, are generated by your definitions for the following queries in Prolog: ?- brother(X,Y). ?- cousin(X,Y). ?- grandson(X,Y). ?- descendent(X,Y).</p>		

9	Write a program to solve Tower of Hanoi problem using Prolog.		
10	Write a program to solve N-Queens problem using Prolog.		

## Practical-1

---

**Aim : -**

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

---

**Code:-**

```
from datetime import datetime
now = datetime.now()
board = [ ' ' for x in range(10) ]

def insertLetter(letter, position):
    board[position] = letter

def spaceIsFree(position):
    if board[position] == ' ':
        return True
    else:
        return False

def printBoard(board):
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print('-----')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print('-----')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print('-----')

def isWinner(bd, ltr):
    return ((bd[1] == ltr and bd[2] == ltr and bd[3] == ltr ) or
            (bd[4] == ltr and bd[5] == ltr and bd[6] == ltr ) or
            (bd[7] == ltr and bd[8] == ltr and bd[9] == ltr ) or
            (bd[1] == ltr and bd[4] == ltr and bd[7] == ltr ) or
            (bd[2] == ltr and bd[5] == ltr and bd[8] == ltr ) or
            (bd[3] == ltr and bd[6] == ltr and bd[9] == ltr ) or
            (bd[1] == ltr and bd[5] == ltr and bd[9] == ltr ) or
            (bd[3] == ltr and bd[5] == ltr and bd[7] == ltr ))
```



```
def playerMove():
    run= True
    while run:
        inputMove = input('Enter 1-9 to as position you want to put X: ')
        try:
            move= int(inputMove)
            if move > 0 and move < 10:
                if spaceIsFree(move):
                    run = False
                    insertLetter('X',move)
                else:
                    print('You select already taken position!')
            else:
                print('Enter number between 1-9')
        except:
            print('Enter digit!')

def compMove():
    pMoves = [x for x, letter in enumerate(board) if letter == ' ' and x != 0 ]
    move = 0
    for let in ['O','X']:
        for i in pMoves:
            boardCopy = board[:]
            boardCopy[i]=let
            if isWinner(boardCopy, let):
                move = i
                return move
    openCorner = []
    for i in pMoves:
        if i in [1,3,7,9]:
            openCorner.append(i)

    if len(openCorner)>0:
        move = selectRandom(openCorner)
        return move

    if 5 in pMoves:
        move = 5
```

```
    return move
    openEdges = []
    for i in pMoves:
        if i in [2,4,6,8]:
            openEdges.append(i)

    if len(openEdges)>0:
        move = selectRandom(openEdges)
    return move
def selectRandom(moveList):
    import random
    ln = len(moveList)
    r = random.randrange(0,ln)
    return moveList[r]
def isBoardFull(board):
    if board.count(' ')>1:
        return False
    else:
        return True
def main():
    print('Enrollment No: 190130107041')
    print('Practical1')
    print('Write a program to implement a Tic-Tac-Toe game problem')
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
    print("date and time =", dt_string)
    printBoard(board)
    while not(isBoardFull(board)):
        if not(isWinner(board,'O')):
            playerMove()
            printBoard(board)
        else:
            print('O is the winner!')
            break
        if not(isWinner(board,'X')):
            move = compMove()
            if move==0 :
                print('Tie Game!')
            else:
                insertLetter('O', move)
                print('O placed at position: ',move)
```

```
        printBoard(board)
    else:
        print('X is the winner!')
        break
    if isBoardFull(board):
        print('No Wins')
        break
main()
```

## Output:

```
Enrollment No: 190130107041
Name: Niraj Italiya
Practical1
Write a program to implement a Tic-Tac-Toe game problem
date and time = 06/11/2022 21:11:01

| | |
-----
| | |
-----
| | |
-----
Enter 1-9 to as position you want to put X: 1
X | | 
-----
| | |
-----
| | |
-----
O placed at position: 9
X | | O
-----
| | |
-----
| | O
-----
Enter 1-9 to as position you want to put X: 5
X | | 
-----
| X | 
-----
| | O
-----
O placed at position: 3
X | | O
-----
| X | 
-----
| | O
-----
Enter 1-9 to as position you want to put X: 9
You select already taken position!
Enter 1-9 to as position you want to put X: 4
X | | O
-----
X | X | 
-----
| | O
-----
O placed at position: 6
X | | O
-----
X | X | O
-----
| | O
-----
O is the winner!
```

## Practical-2

---

**Aim:-**

**Write a program to implement BFS (for 8 puzzle problem or Water Jug problem or any AI search problem) .**

---

**Code:-**

```
from collections import defaultdict
from datetime import datetime

now = datetime.now()

visited = defaultdict(lambda: False)
j1, j2, t = 5, 2, 3
def waterJug(x,y):
    global j1, j2, t
    if (x == t and y == 0) or (y == t and x == 0):
        print('(',x,",",y,")")
        return True
    if visited[(x,y)] == False:
        print('(',x,",",y,")")
        visited[(x,y)] = True

        return (waterJug(x,0) or waterJug(0,y)
                or waterJug(j1,y) or waterJug(x,j2)
                or waterJug(x+min(y,(j1-x)),y-min(y,(j1-x)))
                or waterJug(x-min(x,(j2-y)),y+min(x,(j2-y))))
    else:
        return False
def main():
    j1=0
    print('Name : Italiya Niraj')
    print('Enrollment No: 190130107041')
    print('Practical2')
```

```
print('Write a program to implement BFS (for 8 puzzle problems or Water  
Jug problems or any AI search problem) ')  
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  
print("date and time =", dt_string)  
waterJug(0,0)  
  
main()
```

## **Output:**

```
Name: Italiya Niraj  
Enrollment No: 190130107041  
Practical2  
Write a program to implement BFS (for 8 puzzle problems or Water Jug problems or any AI search problem)  
date and time = 06/11/2022 21:19:10  
( 0 , 0 )  
( 5 , 0 )  
( 5 , 2 )  
( 0 , 2 )  
( 2 , 0 )  
( 2 , 2 )  
( 4 , 0 )  
( 4 , 2 )  
( 5 , 1 )  
( 0 , 1 )  
( 1 , 0 )  
( 1 , 2 )  
( 3 , 0 )
```

## Practical-3

---

### Aim :-

**Write a program to implement DFS (for 8 puzzle problem or Water Jug problem or any AI search problem)**

---

### Code:-

```
from datetime import datetime
now = datetime.now()

import copy
ip = [[1,2,3],[4,-1,5],[6,7,8]]
op = [[1,2,3],[6,4,5],[-1,7,8]]

def move(temp,mv):
    if mv == "up":
        for i in range(3):
            for j in range(3):
                if(temp[i][j] == -1):
                    if i != 0:
                        temp[i][j] = temp[i-1][j]
                        temp[i-1][j] = -1
                    return temp

    if mv == "down":
        for i in range(3):
            for j in range(3):
                if(temp[i][j] == -1):
                    if i != 2:
                        temp[i][j] = temp[i+1][j]
                        temp[i+1][j] = -1
                    return temp

    if mv == "left":
        for i in range(3):
```

```
        for j in range(3):
            if(temp[i][j] == -1):
                if j != 0:
                    temp[i][j] = temp[i][j-1]
                    temp[i][j-1] = -1
                return temp

    if mv == "right":
        for i in range(3):
            for j in range(3):
                if(temp[i][j] == -1):
                    if j != 2:
                        temp[i][j] = temp[i][j+1]
                        temp[i][j+1] = -1
                    return temp

def dfs_8puz():
    global ip, op
    pathcost = 0

    stack = []
    ipx = [ip,"none"]
    stack.append(ipx)

    while(True):
        puzzle = stack.pop(0)
        pathcost = pathcost + 1
        print(str(puzzle[1])+ "--->" + str(puzzle[0]))
        if(puzzle[0] == op):
            print('Done!!!')
            print("Path Cost -->" + str(pathcost-1))
            break
        else:
            if puzzle[1] != "down":
                temp = copy.deepcopy(puzzle[0])
                up = move(temp,"up")
                if up != puzzle[0]:
                    upx = [up,"up"]
                    stack.insert(0,upx)
```



```
    if puzzle[1] != "UP":
        temp = copy.deepcopy(puzzle[0])
        down = move(temp,"down")
        if down != puzzle[0]:
            downx = [down,"down"]
            stack.insert(0,downx)

    if puzzle[1] != "left":
        temp = copy.deepcopy(puzzle[0])
        right = move(temp,"right")
        if right != puzzle[0]:
            rightx = [right,"right"]
            stack.insert(0,rightx)
    if puzzle[1] != "right":
        temp = copy.deepcopy(puzzle[0])
        left = move(temp,"left")
        if left != puzzle[0]:
            leftx = [left,"left"]
            stack.insert(0,leftx)

def main():
    print('Name: Niraj Italiya')
    print('Enrollment No: 190130107041')
    print('Practical3')
    print('Write a program to implement DFS (for 8 puzzle problems or Water Jug
problems or any AI search problem) ')
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
    print("date and time =", dt_string)
    print('Answer')
    dfs_8puz()

main()
```

**Output:**

```
Name: Niraj Italiya
Enrollment No: 190130107041
Practical3
Write a program to implement DFS (for 8 puzzle problems or Water Jug problems or any AI search problem)
date and time = 06/11/2022 21:26:24
Answer
none--->[[1, 2, 3], [4, -1, 5], [6, 7, 8]]
left--->[[1, 2, 3], [-1, 4, 5], [6, 7, 8]]
down--->[[1, 2, 3], [6, 4, 5], [-1, 7, 8]]
Done!!!
Path Cost -->2
```

## Practical-4

---

### Aim :-

**Write a program to implement Single Player Game (Using any Heuristic Function)**

---

### Code:-

```
from datetime import datetime
now = datetime.now()

import random
import math
print('Name: Niraj Italiya')
print('Enrollment No: 190130107041')
print('Practical4')
print('Write a program to implement Single Player Game')
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("date and time =", dt_string)
print('Range of Random Numbers: 0 - 1000')
x = random.randint(0, 1001)
print("\n\tYou have only ", round(math.log(1000 + 1, 2)), "chances to guess the integer!\n")
count = 0
while count < math.log(1000 + 1, 2):
    count += 1
    guess = int(input("Guess a number:-"))
    if x == guess:
        print("Congratulations you did it in ", count, "step")
        break
    elif x > guess:
        print("You guessed too small!")
    elif x < guess:
        print("You Guessed too high!")
    if count >= math.log(1000 + 1, 2):
        print("\nThe number is %d" % x)
```

```
print("\tBetter luck next time")
```

### **Output:**

```
Name: Niraj Italiya
Enrollment No: 190130107041
Practical4
Write a program to implement Single Player Game
date and time = 06/11/2022 21:29:31
Range of Random Numbers: 0 - 1000

        You have only 10 chances to guess the integer!

Guess a number:-500
You guessed too small!
Guess a number:-800
You Guessed too high!
Guess a number:-700
You Guessed too high!
Guess a number:-600
You Guessed too high!
Guess a number:-550
You Guessed too high!
Guess a number:-525
You guessed too small!
Guess a number:-535
You Guessed too high!
Guess a number:-529
You guessed too small!
Guess a number:-532
You Guessed too high!
Guess a number:-531
Congratulations you did it in 10 step|
```

## Practical-5

---

### **Aim : -**

**Write a program to Implement A\* Algorithm**

---

### **Code:**

```
from datetime import datetime
now = datetime.now()

import copy

print('Name: Italiya Niraj')
print('Enrollment No: 190130107041')
print('Practical5')
print('Write a program to Implement A* Algorithm. ')
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("date and time =", dt_string)
def astar(start,end):
    #initialise param
    open = set(start)
    close = set()
    g = {}
    parent = {}
    g[start] = 0
    parent[start] = start

    while len(open) > 0:
        n = None

        for i in open.copy():
            if n == None or (g[i] + h(i)) < (g[n] + h(n)):
                n = i

            if n == end or graph_nodes[n] == None:
                pass
```

```
    else:
        for(m,weight) in get_neighbors(n):
            if m not in open and m not in close:
                open.add(m)
                parent[m] = n
                g[m] = g[n] + weight
            else:
                if g[m] > g[n] + weight:
                    g[m] = g[n] + weight
                    parent[m] = n

                if m in close:
                    close.remove(m)
                    open.add(m)

    if n == None:
        print("Path doesn't exist")
        return None
    if n == end:
        path = []

        while parent[n] != n:
            path.append(n)
            n = parent[n]

        path.append(start)
        path.reverse()

        print('Path found: {}'.format(path))
        return path

    open.remove(n)
    close.add(n)
    print("Path doesn't exist")
    return None

def get_neighbors(v):
    if v in graph_nodes:
        return graph_nodes[v]
    else:
```

```
        return None

def h(n):
    H_dist={
        'A': 9,
        'B': 7,
        'C': 6,
        'D': 99,
        'E': 0,
        'G': 19,
    }
    return H_dist[n]

graph_nodes = {
    'A': [('B',2), ('E',5)],
    'B': [('C',8), ('G',3)],
    'C': None,
    'E': [('D',1)],
    'D': [('G',2)],
}
astar('A','G')
```

## **Output:**

```
Name: Italiya Niraj
Enrollment No: 190130107041
Practical5
Write a program to Implement A* Algorithm.
date and time = 06/11/2022 21:35:17
Path found: ['A', 'B', 'G']

['A', 'B', 'G']
```

## Practical-6

---

### Aim:-

**Write a program to implement mini-max algorithm for any game development.**

---

### Code:-

```
from datetime import datetime
now = datetime.now()

print('Name: Italiya Niraj ')
print('Enrollment No: 190130107041')
print('Practical6')
print('Write a program to implement a minimax algorithm for any game
development ')
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
print("date and time =", dt_string)

player, opponent = 'x', 'o'
def isMovesLeft(board):
    for i in range(3):
        for j in range(3):
            if (board[i][j] == '_'):
                return True
    return False

def evaluate(b):
    for row in range(3):
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2]):
            if (b[row][0] == player):
                return 10
            elif (b[row][0] == opponent):
                return -10
    for col in range(3):
        if (b[0][col] == b[1][col] and b[1][col] == b[2][col]):
            if (b[0][col] == player):
```



```
        return 10
    elif (b[0][col] == opponent):
        return -10
    if (b[0][0] == b[1][1] and b[1][1] == b[2][2]):
        if (b[0][0] == player):
            return 10
        elif (b[0][0] == opponent):
            return -10
    if (b[0][2] == b[1][1] and b[1][1] == b[2][0]):
        if (b[0][2] == player):
            return 10
        elif (b[0][2] == opponent):
            return -10
    return 0

def minimax(board, depth, isMax):
    score = evaluate(board)
    if (score == 10):
        return score
    if (score == -10):
        return score
    if (isMovesLeft(board) == False):
        return 0
    if (isMax):
        best = -1000
        for i in range(3):
            for j in range(3):
                if (board[i][j] == '_'):
                    board[i][j] = player
                    best = max(best, minimax(board, depth + 1, not isMax))
                    board[i][j] = '_'
        return best
    else:
        best = 1000
        for i in range(3):
            for j in range(3):
                if (board[i][j] == '_'):
                    board[i][j] = opponent
                    best = min(best, minimax(board, depth + 1, not isMax))
                    board[i][j] = '_'
```

```
        return best

def findBestMove(board):
    bestVal = -1000
    bestMove = (-1, -1)
    for i in range(3):
        for j in range(3):
            if (board[i][j] == '_'):
                board[i][j] = player
                moveVal = minimax(board, 0, False)
                board[i][j] = '_'
                if (moveVal > bestVal):
                    bestMove = (i, j)
                    bestVal = moveVal
    print("Value :", bestVal)
    print()
    return bestMove

board = [['x', 'o', 'x'], ['o', 'o', 'x'], ['_', '_', '_']]
bestMove = findBestMove(board)
print("The Optimal Move is :", "Row:", bestMove[0], " Col:", bestMove[1])
print("Before Move:")
print(board)
board[bestMove[0]][bestMove[1]]='x'
print("After Move:")
print(board)
```

## **Output:**

```
Name: Italiya Niraj
Enrollment No: 190130107041
Practical6
Write a program to implement a minimax algorithm for any game development
date and time = 06/11/2022 21:44:07
Value : 10

The Optimal Move is : Row: 2  Col: 2
Before Move:
[['x', 'o', 'x'], ['o', 'o', 'x'], ['_', '_', '_']]
After Move:
[['x', 'o', 'x'], ['o', 'o', 'x'], ['_', '_', 'x']]
```

## Practical-7&8

---

### Aim:-

7. Assume given a set of facts of the form `father(name1,name2)` (`name1` is the father of `name2`).

### And

8. Define a predicate `brother(X,Y)` which holds iff `X` and `Y` are brothers.

Define a predicate `cousin(X,Y)` which holds iff `X` and `Y` are cousins.

Define a predicate `grandson(X,Y)` which holds iff `X` is a grandson of `Y`.

Define a predicate `descendent(X,Y)` which holds iff `X` is a descendent of `Y`.

Consider the following genealogical tree:

`father(a,b).`

`father(a,c).`

`father(b,d).`

`father(b,e).`

`father(c,f).`

Say which answers, and in which order, are generated by your definitions for the following queries in Prolog:

?- `brother(X,Y).`

?- `cousin(X,Y).`

?- `grandson(X,Y).`

?- `descendent(X,Y).`

---

### Code:

```
father(a,b).
```

```
father(a,c).
```

```
father(b,d).
```

```
father(b,e).
```

```
father(c,f).
```

```
brother(X,Y) :- father(Z,X), father(Z,Y), not(X=Y).
```

```
cousin(X,Y) :- father(Z,X), father(W,Y), brother(Z,W).
```

```
grandson(X,Y) :- father(Z,X), father(Y,Z).
```

```
descendent(X,Y) :- father(Y,X).
```

```
descendent(X,Y) :- father(Z,X), descendent(Z,Y).
```

```
:- initialization(main).
```

```
main :-
```

```
write('enrollment : 190130107041'),nl,  
write('Practical : 07+08 '),nl,nl,  
write(Niraj Italiya'),nl,nl,  
write('Assume given a set of facts of the form father(name1,name2) (name1 is the  
father of name2).  
Define a predicate brother(X,Y) which holds iff X and Y are brothers.  
Define a predicate cousin(X,Y) which holds iff X and Y are cousins.  
Define a predicate grandson(X,Y) which holds iff X is a grandson of Y. Define a  
predicate descendent(X,Y) which holds iff X is a descendent of Y.  
Consider the following genealogical tree:  
father(a,b). father(a,c). father(b,d). father(b,e). father(c,f).  
Say which answers, and in which order, are generated by your definitions for the  
following queries in Prolog:  
?- brother(X,Y).  
?- cousin(X,Y).  
?- grandson(X,Y).  
?- descendent(X,Y)  
)',nl,nl.
```

## **Output:**

```

grandson(X,Y).

enrollment : 190130107041
Practical : 07+08

Niraj Italiya

Assume given a set of facts of the form father(name1,name2)
(name1 is the father of name2). Define a predicate
brother(X,Y) which holds iff X and Y are brothers. Define a
predicate cousin(X,Y) which holds iff X and Y are cousins.
Define a predicate grandson(X,Y) which holds iff X is a
grandson of Y. Define a predicate descendent(X,Y) which
holds iff X is a descendent of Y. Consider the following
genealogical tree: father(a,b). father(a,c). father(b,d).
father(b,e). father(c,f). Say which answers, and in which
order, are generated by your definitions for the following
queries in Prolog: ?- brother(X,Y). ?- cousin(X,Y). ?-
grandson(X,Y). ?- descendent(X,Y)

X = d,
Y = a
X = e,
Y = a
X = f,
Y = a

```

```

cousin(X,Y).

enrollment : 190130107041
Practical : 07+08

Niraj Italiya

Assume given a set of facts of the form father(name1,name2)
(name1 is the father of name2). Define a predicate
brother(X,Y) which holds iff X and Y are brothers. Define a
predicate cousin(X,Y) which holds iff X and Y are cousins.
Define a predicate grandson(X,Y) which holds iff X is a
grandson of Y. Define a predicate descendent(X,Y) which
holds iff X is a descendent of Y. Consider the following
genealogical tree: father(a,b). father(a,c). father(b,d).
father(b,e). father(c,f). Say which answers, and in which
order, are generated by your definitions for the following
queries in Prolog: ?- brother(X,Y). ?- cousin(X,Y). ?-
grandson(X,Y). ?- descendent(X,Y)

X = d,
Y = f
X = e,
Y = f
X = f,
Y = d
X = f,
Y = e
false

```

```

brother(X,Y).
enrollment : 190130107041
Practical : 07+08

Niraj Italiya

Assume given a set of facts of the form father(name1,name2)
(name1 is the father of name2). Define a predicate
brother(X,Y) which holds iff X and Y are brothers. Define a
predicate cousin(X,Y) which holds iff X and Y are cousins.
Define a predicate grandson(X,Y) which holds iff X is a
grandson of Y. Define a predicate descendant(X,Y) which
holds iff X is a descendant of Y. Consider the following
genealogical tree: father(a,b). father(a,c). father(b,d).
father(b,e). father(c,f). Say which answers, and in which
order, are generated by your definitions for the following
queries in Prolog: ?- brother(X,Y). ?- cousin(X,Y). ?-
grandson(X,Y). ?- descendant(X,Y)

X = b,
Y = c
X = c,
Y = b
X = d,
Y = e
X = e,
Y = d
false

```

enrollment : 190130107041  
Practical : 07+08

Niraj Italiya

Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2). Define a predicate brother(X,Y) which holds iff X and Y are brothers. Define a predicate cousin(X,Y) which holds iff X and Y are cousins. Define a predicate grandson(X,Y) which holds iff X is a grandson of Y. Define a predicate descendant(X,Y) which holds iff X is a descendant of Y. Consider the following genealogical tree: father(a,b). father(a,c). father(b,d). father(b,e). father(c,f). Say which answers, and in which order, are generated by your definitions for the following queries in Prolog: ?- brother(X,Y). ?- cousin(X,Y). ?- grandson(X,Y). ?- descendant(X,Y)

```

X = b,
Y = a
X = c,
Y = a
X = d,
Y = b
X = e,
Y = b
X = f,
Y = c
X = d,
Y = a
X = e,
Y = a
X = f,
Y = a
false

```

## Practical-9

---

### Aim:-

Write a program to solve Tower of Hanoi problem using Prolog.

---

### Code:-

```
move(1,X,Y,_):-
  write('Move top disk from '), write(X), write(' to '), write(Y), nl.
move(N,X,Y,Z):-
  N>1,
  M is N-1,
  move(M,X,Z,Y),
  move(1,X,Y,_),
  move(M,Z,Y,X).

:- initialization(main).
main :-
  write('enrollment : 190130107041'),nl,
  write('name : Niraj Italiya'),nl,
  write('Practical : 09 '),nl,
  write('program to solve tower of hanoi'),nl,nl,
  move(3,source,target,auxiliary).
```

**Output:-**

enrollment : 190130107041

name : Niraj Italiya

Practical : 09

program to solve tower of hanoi

Move top disk from source to target

Move top disk from source to auxiliary

Move top disk from target to auxiliary

Move top disk from source to target

Move top disk from auxiliary to source

Move top disk from auxiliary to target

Move top disk from source to target



## Practical-10

### Aim:-

Write a program to solve N-Queens problem using Prolog.

### Code:-

```
:- use_rendering(chess).

queens(N, Queens) :-
    length(Queens, N),
    board(Queens, Board, 0, N, _, _),
    queens(Board, 0, Queens).

board([], [], N, N, _, _).
board([_|Queens], [Col-Vars|Board], Col0, N, [_|VR], VC) :-
    Col is Col0+1,
    functor(Vars, f, N),
    constraints(N, Vars, VR, VC),
    board(Queens, Board, Col, N, VR, [_|VC]).

constraints(0, _, _, _) :- !.
constraints(N, Row, [R|Rs], [C|Cs]) :-
    arg(N, Row, R-C),
    M is N-1,
    constraints(M, Row, Rs, Cs).

queens([], _, []).
queens([C|Cs], Row0, [Col|Solution]) :-
    Row is Row0+1,
    select(Col-Vars, [C|Cs], Board),
    arg(Row, Vars, Row-Row),
    queens(Board, Row, Solution).

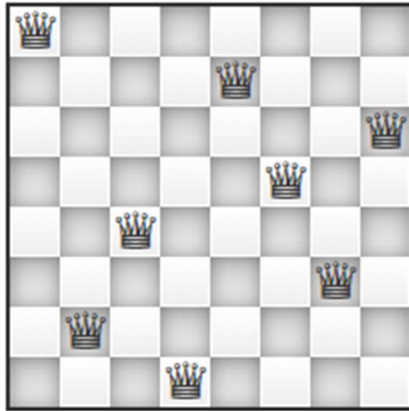
:- initialization(main).
main :-
    write('enrollment : 190130107047'),nl,
```

```
write('name : Kansara Bhavya Nimeshkumar'),nl,  
write('Practical : 10 '),nl,  
write('program to solve N queen problem'),nl,nl,  
queens(8, Queens).
```

## **Output:**

```
enrollment : 190130107041  
name : Niraj Italiya  
Practical : 10  
program to solve N queen problem
```

**Queens =**



## Practical-11

---

### Aim:-

Write a program to solve 8 puzzle problem using Prolog.

---

### Code:-

```
:- initialization(main).
main :- write('Enrollment : 190130107047'),nl,
        write('Name : Kansara Bhavya Nimeshkumar'),nl,
        write('Practical : 11'),nl,
        write('Aim:program to solve 8 puzzle problem'),nl,nl.
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).
%Determines whether Current and Destination tiles are a valid move.
is_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 -
Y1).
% This predicate produces the plan. Once the Goal list is a subset
% of the current State the plan is complete and it is written to
% the screen using write_sol/1.
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).
% The problem has three operators.
% 1st arg = name
% 2nd arg = preconditions
% 3rd arg = delete list
% 4th arg = add list.
% Tile can move to new position only if the destination tile is empty & Manhattan
distance = 1.
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)]).
% Utility predicates.
% Check if first list is a subset of the second
is_subset([H|T], Set):-
member(H, Set),
is_subset(T, Set).
is_subset([], _).
% Remove all elements of 1st list from second to create third.
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:**

Enrollment : 190130107041

Name : Niraj Italiya

Practical : 11

Aim: program to solve 8 puzzle problem

Initial state:

*at*(tile7,9)

*at*(tile1,8)

*at*(tile5,7)

*at*(tile6,6)

*at*(tile2,5)

*at*(empty,4)

*at*(tile8,3)

*at*(tile3,2)

*at*(tile4,1)

Goal state:

[*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)]

## Practical-12

---

### Aim:-

Write a program to solve travelling salesman problem using Prolog

---

### Code:-

```
road(birmingham,bristol, 9).
road(london,birmingham, 3).
road(london,bristol, 6).
road(london,plymouth, 5).
road(plymouth,london, 5).
road(portsmouth,london, 4).
road(portsmouth,plymouth, 8).

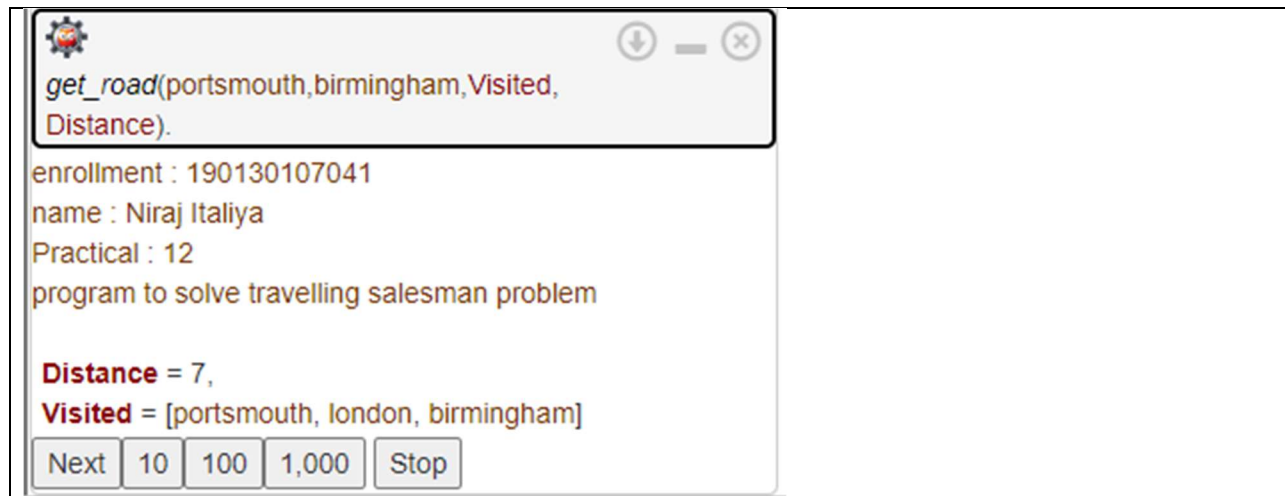
get_road(Start, End, Visited, Result) :-
    get_road(Start, End, [Start], 0, Visited, Result).

get_road(Start, End, Waypoints, DistanceAcc, Visited, TotalDistance) :-
    road(Start, End, Distance),
    reverse([End|Waypoints], Visited),
    TotalDistance is DistanceAcc + Distance.

get_road(Start, End, Waypoints, DistanceAcc, Visited, TotalDistance) :-
    road(Start, Waypoint, Distance),
    \+ member(Waypoint, Waypoints),
    NewDistanceAcc is DistanceAcc + Distance,
    get_road(Waypoint, End, [Waypoint|Waypoints], NewDistanceAcc, Visited,
    TotalDistance).

:- initialization(main).
main :-
    write('enrollment : 190130107047'),nl,
    write('name : Kansara Bhavya Nimeshkumar'),nl,
    write('Practical : 12 '),nl,
    write('program to solve travelling salesman problem'),nl,nl.
```

## Output:



The screenshot shows a Java Swing window with a title bar containing a gear icon, a download icon, a minimize icon, and a close icon. The window title is `get_road(portsmouth,birmingham,Visited,Distance)`. The main content area displays the following text:

```
enrollment : 190130107041  
name : Niraj Italiya  
Practical : 12  
program to solve travelling salesman problem  
  
Distance = 7,  
Visited = [portsmouth, london, birmingham]
```

At the bottom of the window, there is a row of five buttons: "Next", "10", "100", "1,000", and "Stop".