Name : ANKIT KUMAR SAHU

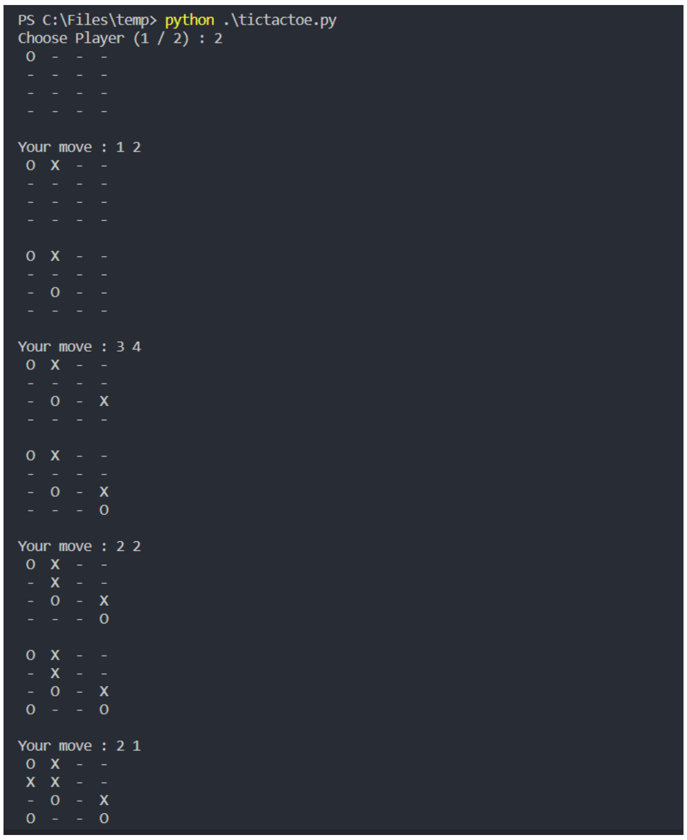
Reg No : 20BAI1005

Lab – 8

ALPHA BETA PRUNING

```python
import numpy as np
from copy import copy

rows=4
cols=4

board = np.zeros((rows,cols))

inf=1e6
neg_inf=-1e6

def print_board():
    for i in range(0,rows):
        for j in range(0,cols):
            if board[i,j]==0:
                print(' - ', end='')
            elif board[i,j]==1:
                print(' X ', end='')
            else:
                print(' O ', end='')
        print()


htable = np.zeros((rows+1,cols+1))
n = rows+cols+2

for i in range(0,rows+1):
    htable[i,0]=10**i
    htable[0,i]=-10**i

winning_pos =
np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11],[12,13,14,15],[0,4,8,12],[1,5,9,13],[2,6,10,14],
[3,7,11,15], [0,5,10,15],[3,6,9,12]])

def state_utility(state):
    stateCopy=copy(state.ravel())
    heuristic=0
    for i in range(0,n):
        maxp=0
        minp=0
        for j in range(0, rows):
            if stateCopy[winning_pos[i,j]]==2:
                maxp+=1
            elif stateCopy[winning_pos[i,j]]==1:
                minp+=1
        heuristic+=htable[maxp][minp]
    return heuristic

def minimax(state,alpha,beta,maximizing,depth,maxp,minp):
    if depth==0:
        return state_utility(state),state
    rowsLeft,columnsLeft=np.where(state==0)
    returnState=copy(state)
    if rowsLeft.shape[0]==0:
        return state_utility(state),returnState
    if maximizing==True:
        utility=neg_inf
        for i in range(0,rowsLeft.shape[0]):
            nextState=copy(state)
            nextState[rowsLeft[i],columnsLeft[i]]=maxp
            Nutility,Nstate=minimax(nextState,alpha,beta,False,depth-1,maxp,minp)
            if Nutility > utility:
                utility=Nutility
                returnState=copy(nextState)
            if utility>alpha:
                alpha=utility
            if alpha >=beta :
                break
        return utility,returnState
    else:
        utility=inf
        for i in range(0,rowsLeft.shape[0]):
            nextState=copy(state)
            nextState[rowsLeft[i],columnsLeft[i]]=minp
            Nutility,Nstate=minimax(nextState,alpha,beta,True,depth-1,maxp,minp)
            if Nutility < utility:
                utility=Nutility
                returnState=copy(nextState)
            if utility< beta:
                beta=utility
            if alpha >=beta:
                break
        return utility,returnState

def checkGameOver(state):
    stateCopy=copy(state)
    value=state_utility(stateCopy)
    if value >=1000:
        return 1
    return -1

def main():
    num=input('Choose Player (1 / 2) : ')
    num = int(num)
    value = 0
    global board
    for turn in range(0,rows*cols):
        if (turn+num)%2==1:
            r,c=[int(x) for x in input('Your move : ').split(' ')]
            board[r-1,c-1]=1
            print_board()
            value=checkGameOver(board)
            if value==1:
                print('Congrats! You Won')
                exit()
            print()
        else:
            state=copy(board)
            value,nextState=minimax(state,neg_inf,inf,True,2,2,1)
            board=copy(nextState)
            print_board()
            print()
            value=checkGameOver(board)
            if value==1:
                print('Opponent Won')
                exit()
    print('DRAW')
main()
```

OUTPUT



```
PS C:\Files\temp> python .\tictactoe.py
Choose Player (1 / 2) : 2
 O  -  -  -
 -  -  -  -
 -  -  -  -
 -  -  -  -

Your move : 1 2
 O  X  -  -
 -  -  -  -
 -  -  -  -
 -  -  -  -

 O  X  -  -
 -  -  -  -
 -  O  -  -
 -  -  -  -

Your move : 3 4
 O  X  -  -
 -  -  -  -
 -  O  -  X
 -  -  -  -

 O  X  -  -
 -  -  -  -
 -  O  -  X
 -  -  -  O

Your move : 2 2
 O  X  -  -
 -  X  -  -
 -  O  -  X
 -  -  -  O

 O  X  -  -
 -  X  -  -
 -  O  -  X
 O  -  -  O

Your move : 2 1
 O  X  -  -
 X  X  -  -
 -  O  -  X
 O  -  -  O
```