RA1911030010081

A SOWJANYA

AI EXPERIMENT – 1

**CAMEL AND BANANA PROBLEM:**

CODE:

dp = [[-1 for i in range(3001)] for j in range(1001)]

def recBananaCnt(A, B, C):

if (B <= A):

return 0

if (B <= C):

return B - A

if (A == 0):

return B

if (dp[A][B] != -1):

return dp[A][B]

maxCount = -2\*\*32

tripCount = ((2 \* B) // C) - 1 if(B % C == 0 ) else ((2 \* B) // C) + 1

for i in range(1,A+1):

curCount = recBananaCnt(A - i, B - tripCount \* i, C)

if (curCount > maxCount):

maxCount = curCount

dp[A][B] = maxCount

return maxCount

def maxBananaCnt(A, B, C):

return recBananaCnt(A, B, C)

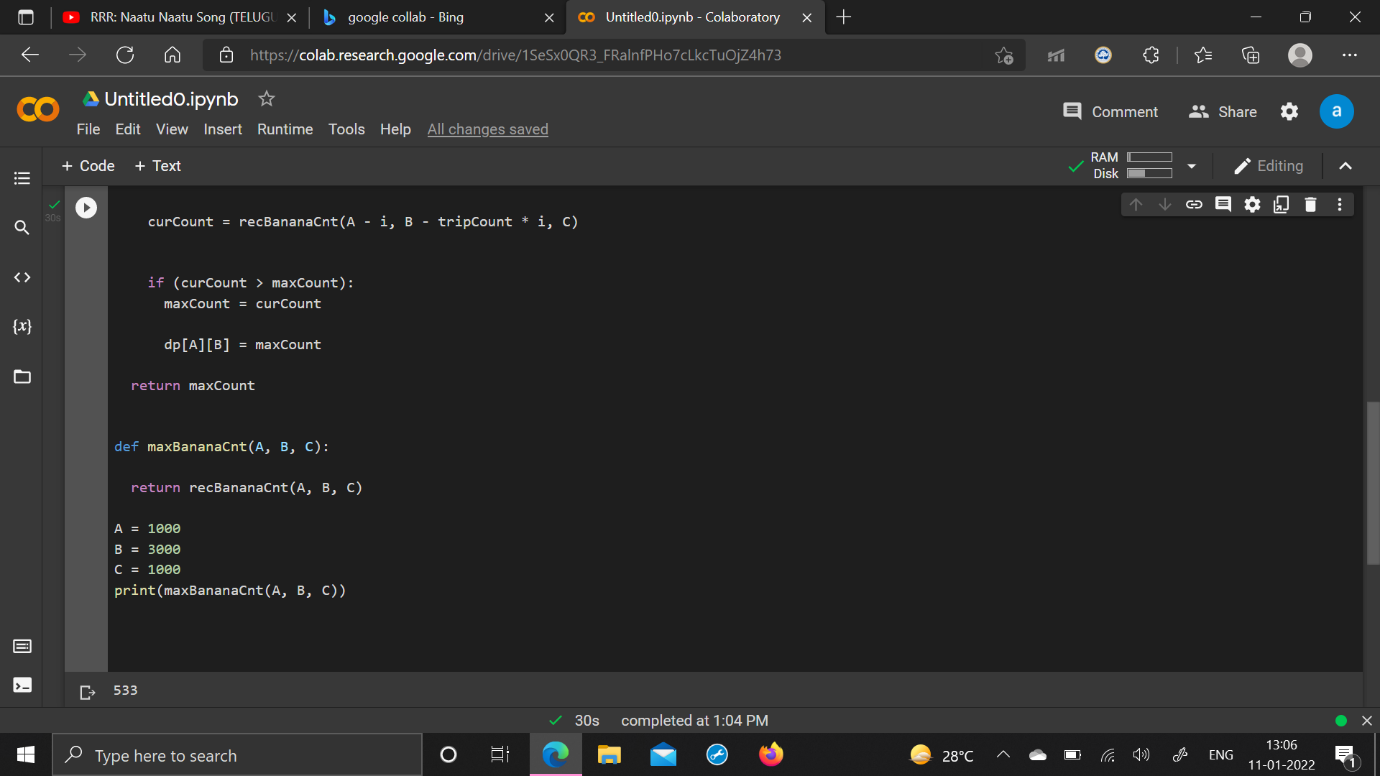
A = 1000

B = 3000

C = 1000

print(maxBananaCnt(A, B, C))

OUTPUT:



**TIC TAC TOE PROBLEM:**

CODE:

import random

class TicTacToe:

def \_\_init\_\_(self):

self.board = []

def create\_board(self):

for i in range(3):

row = []

for j in range(3):

row.append('-')

self.board.append(row)

def get\_random\_first\_player(self):

return random.randint(0, 1)

def fix\_spot(self, row, col, player):

self.board[row][col] = player

def is\_player\_win(self, player):

win = None

n = len(self.board)

# checking rows

for i in range(n):

win = True

for j in range(n):

if self.board[i][j] != player:

win = False

break

if win:

return win

# checking columns

for i in range(n):

win = True

for j in range(n):

if self.board[j][i] != player:

win = False

break

if win:

return win

# checking diagonals

win = True

for i in range(n):

if self.board[i][i] != player:

win = False

break

if win:

return win

win = True

for i in range(n):

if self.board[i][n - 1 - i] != player:

win = False

break

if win:

return win

return False

for row in self.board:

for item in row:

if item == '-':

return False

return True

def is\_board\_filled(self):

for row in self.board:

for item in row:

if item == '-':

return False

return True

def swap\_player\_turn(self, player):

return 'X' if player == 'O' else 'O'

def show\_board(self):

for row in self.board:

for item in row:

print(item, end=" ")

print()

def start(self):

self.create\_board()

player = 'X' if self.get\_random\_first\_player() == 1 else 'O'

while True:

print(f"Player {player} turn")

self.show\_board()

# taking user input

row, col = list(

map(int, input("Enter row and column numbers to fix spot: ").split()))

print()

# fixing the spot

self.fix\_spot(row - 1, col - 1, player)

# checking whether current player is won or not

if self.is\_player\_win(player):

print(f"Player {player} wins the game!")

break

# checking whether the game is draw or not

if self.is\_board\_filled():

print("Match Draw!")

break

# swapping the turn

player = self.swap\_player\_turn(player)

# showing the final view of board

print()

self.show\_board()

# starting the game

tic\_tac\_toe = TicTacToe()

tic\_tac\_toe.start()

OUTPUT:

