Agricultural Development Trust’s

Shardabai Pawar Art,Commerce & Science College Shardanagar,

Malegaon (Bk), Tal-Baramati, Dist- Pune 413 115.

Department Of Computer Science

CERTIFICATE

Seat No:-

Date:-

This is to certify that, the work entered in this journal is the work of Mr. / Miss____. Who has worked for the I / II / III / IV semester of the MSc (Comp.Sci) Year 2025- 2026 in the College. He / She has completed work as prescribed by Savitribai Phule Pune University, Pune.

Practical in-Charge **Head of Department**

Internal Examiner **External Examiner**

INDEX

Subject:-Lab course on CS-502-MJ(Artificial Intelligence) **Seat No:-**

16	Write a Python program to solve water jug problem.		
17	Write a Python program to simulate 4-Queens problem		
18	Write a Python program to implement Mini-Max Algorithm.		
19	Write a Python program to simulate 8-Queens problem.		
20	Write a python program to sort the sentence in alphabetical order?		
21	Write a Python program to simulate n-Queens problem		
22	Write a Program to Implement Monkey Banana Problem using Python		
23	Write a program to implement Iterative Deepening DFS algorithm.		
24	Write a Program to Implement Tower of Hanoi using Python		
25	Python program that demonstrates the hill climbing algorithm to find the maximum of a mathematical function.		
26	Build a bot which provides all the information related to you in college.		
27	Write a python program to remove punctuations from the given string?		
28	Write a Python program for the following Cryptarithmetic problems.GO + TO = OUT		
29	Write a Program to Implement Alpha-Beta Pruning using Python.		
30	Write a Python program for the following Cryptarithmetic problems SEND + MORE = MONEY		
31	Write a Python program for the following Cryptorithmetic problems CROSS+ROADS = DANGER.		

SLIP NO-01

Q.1) Python program that demonstrates the hill climbing algorithm to find the maximum of a mathematical function.(For example f(x) = -x^2 + 4x) [10Marks]

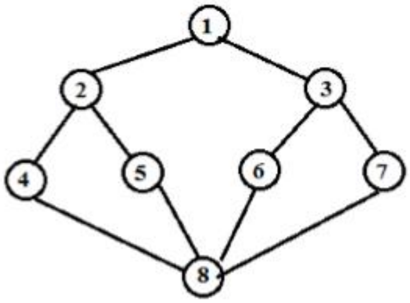
ANS:

```
import numpy as np
def objective_function(x):
    return -x**2+4*x
def hill_climbing(start_x,step_size,num_steps):
    current_x=start_x
    for step in range(num_steps):
        current_value=objective_function(current_x)
        next_x=current_x+step_size
        next_value=objective_function(next_x)
        if next_value > current_value:
            current_x=next_x
        return current_x,objective_function(current_x)
start_x=0.0
step_size=0.1
num_steps=100
best_x,best_value=hill_climbing(start_x,step_size,num_steps)
print(f"Starting point:{start_x:.2f}")
print(f"optimal x:{best_x:.2f}")
print(f"Minimum value:{best_value:.2f}")
```

OUTPUT:

```
Starting point:0.00
optimal x:0.10
Minimum value:0.39
```

Q.2) Write a Python program to implement Depth First Search algorithm. Refer the follow ing graph as an Input for the program. [Initial node=1,Goal node=8]. [20 Marks]



ANS:

```
graph={
    1:[2,3],
    2:[4,5],
    3:[6,7],
    4:[8],
    5:[8],
    6:[8],
    7:[8],
    8:[]
}
goal=8
visited=set()
def dfs(visited,graph,node):
    if node not in visited:
        print(node,"->",end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            if goal in visited:
                break
            else:
                dfs(visited,graph,neighbour)
dfs(visited,graph,1)
```

OUTPUT:

```
1 -> 2 -> 4 -> 8 ->
```

SLIP NO-02

Q.1) Write a python program to generate Calendar for the given month and year?.

[10 Marks]

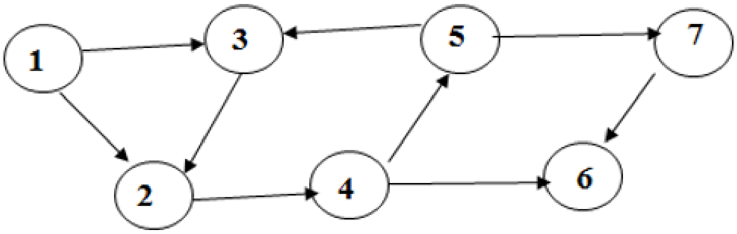
ANS:

```
import calendar

from datetime import datetime

current=datetime.now()

print(calendar.month(current.year, current.month))
```

OUTPUT:

```
    October 2025
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Q.2)Write a Python program to implement Depth First Search algorithm. Refer the following graph as an Input for the program.[Initial node=1,Goal node=7].   [20 Marks]



ANS:

```
graph={
    1:[2,3],
    2:[4],
    3:[2],
    4:[5,6],
    5:[7],
    6:[],
    7:[6]
}
goal=7
visited=set()
def dfs(visited,graph,node):
    if node not in visited:
        print(node,"->",end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            if goal in visited:
```

```
            break
        else:
            dfs(visited,graph,neighbour)
dfs(visited,graph,1)
```

OUTPUT:

```
1 -> 2 -> 4 -> 5 -> 7 ->
```

SLIP NO-03

Q.1) Write a python program to remove punctuations from the given string? .

[10 marks]

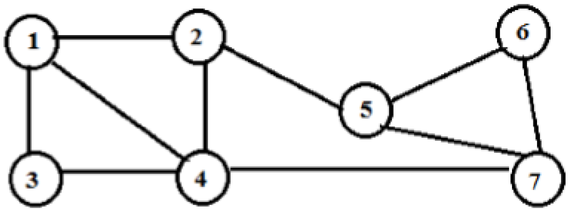
ANS:

```
str=input("Enter a string:")
for c in str:
    if c in "@'!":
        str=str.replace(c,")
print(str)
```

OUTPUT:

```
Enter a string:  ADT's!college
ADTscollege
```

Q.2) Write a Python program to implement Depth First Search algorithm. Refer the following graph as an Input for the program.[Initial node=2,Goal node=7]. [20 Marks]



ANS:

```
graph={
    1:[2,3,4],
    2:[1,4,5],
    3:[1,4],
    4:[1,2,3,7],
    5:[2,6,7],
    6:[5,7],
    7:[]
}
goal=7
visited=set()
def dfs(visited,graph,node):
    if node not in visited:
        print(node,"->",end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            if goal in visited:
                break
            else:
                dfs(visited,graph,neighbour)
dfs(visited,graph,2)
```

OUTPUT:

```
2 -> 1 -> 3 -> 4 -> 7 ->
```

SLIP NO-04

Q.1)Write a program to implement Hangman game using python. [10 Marks]
Description:
Hangman is a classic word-guessing game. The user should guess the word correctly by entering alphabets of the user choice. The Program will get input as single alphabet from the user and it will matchmaking with the alphabets in the original .

ANS:

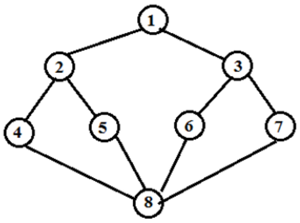
```
import time

import random

name=input("what is your name?")

print("Hello,"+name,"Time to play hangman!")

time.sleep(1)

print("Start guessing...\n")


time.sleep(0.5)

words=['python','programming','treasure','creative','medium','horror']

word=random.choice(words)

guesses=""

turns=10

while turns>0:

    failed=0

    for char in words:

        if char in guesses:

            print(char,end="")

        else:

            print("*",end=""),

            failed+=1
    if failed==0:
        print("\n You Won")
        break
    guess=input("\nguess the character:")
    guesses+=guess
    if guess not in word:
        turns-=1
        print("\nWrong")
        print("\nYou have",+turns,'more guesses')
        if turns==0:
            print("\n You Loss")
```

OUTPUT:

```
what is your name? mcs
Hello,mcs Time to play hangman!
Start guessing...

******
guess the character:p
******
guess the character:y

Wrong

You have 9 more guesses
******
```

```
guess the character:t


Wrong


You have 8 more guesses
******
guess the character:h


Wrong


You have 7 more guesses
******
guess the character:o
******
guess the character:n
python*****
```

Q.2) Write a Python program to implement Breadth First Search algorithm. Refer the following graph as an Input for the program.[Initial node=1,Goal node=8] [20 Marks]



ANS:

```
graph={
    1:[2,3],
    2:[4,5],
    3:[6,7],
    4:[8],
    5:[8],
    6:[8],
    7:[8],
    8:[]
}
visited=[]
queue=[]
goal=8

def bfs(visited,graph,node):
    visited.append(node)
    queue.append(node)
    while queue:
        s=queue.pop(0)
        print(s,"->",end=" ")
        for neighbour in graph[s]:
            if neighbour not in visited:
                visited.append(neighbour)
                queue.append(neighbour)
                if goal in visited:
                    break
bfs(visited,graph,1)
```

OUTPUT:

```
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 ->
```

SLIP NO-05

Q.1) Write a python program to implement Lemmatization using NLTK [10 Marks]

ANS:

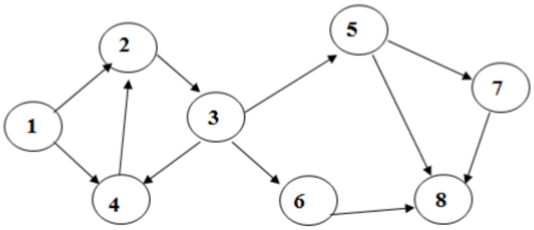
```
lemmatizer=WordNetLemmatizer()
sentence="The children are running towards a better place."
tokens=word_tokenize(sentence)
print(tokens)
tagged_tokens=pos_tag(tokens)
#print(tagged_tokens)
def get_wordnet_pos(tag):
    if tag.startswith("J"):
        return 'a'
    elif tag.startswith("V"):
        return 'v'
    elif tag.startswith("N"):
        return 'n'
    elif tag.startswith("R"):
        return 'r'
    else:
        return 'n'

lemmatized_sentence=[]
for word,tag in tagged_tokens:
    if word.lower()=='are' or word.lower() in ['is','am']:
        lemmatized_sentence.append(word)
    else:
        lemmatized_sentence.append(lemmatizer.lemmatize(word,get_wordnet_pos(tag)))
print("Original sentence:",sentence)
print("lemmatized_sentence:",".join(lemmatized_sentence))
```

OUTPUT:

```
['The', 'children', 'are', 'running', 'towards', 'a', 'better', 'place', '.']
Original sentence: The children are running towards a better place.
lemmatized_sentence: Thechildareruntowardsagoodplace.
```

Q.2) Write a Python program to implement Breadth First Search algorithm. Refer the following graph as an Input for the program.[Initial node=1,Goal node=8]

[20 Marks]



ANS:

```
graph={
    1:[2,4],
    2:[3],
    3:[4,5,6],
    4:[2],
    5:[7,8],
    6:[8],
    7:[8],
    8:[]
}
visited=[]
queue=[]
goal=8


def bfs(visited,graph,node):
    visited.append(node)
    queue.append(node)
    while queue:
        s=queue.pop(0)
        print(s,"->",end=" ")
        for neighbour in graph[s]:
            if neighbour not in visited:
                visited.append(neighbour)
                queue.append(neighbour)
                if goal in visited:
                    break
bfs(visited,graph,1)
```

OUTPUT:

```
1 -> 2 -> 4 -> 3 -> 5 -> 6 -> 7 -> 8 ->
```

SLIP NO-06

Q.1) Write a python program to remove stop words for a given passage from a text file using NLTK?. [10 Marks]

ANS:

```
import nltk

from nltk.corpus import stopwords

with open("input.txt", "r") as file:

    text = file.read()

stop_words = set(stopwords.words("english"))

words = text.split()

filtered_words = [word for word in words if word.lower() not in stop_words]

clean_text = " ".join(filtered_words)

with open("output.txt", "w") as file:

    file.write(clean_text)

print("Stop words removed successfully! Check output.txt")
```

OUTPUT:

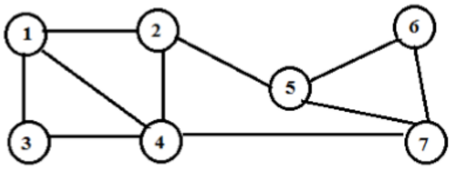
Input.txt

```
This is a sample passage demonstrating the removal of stop words using NLTK in Python.
```

Output.txt

```
sample passage demonstrating removal stop words using NLTK Python.
```

Q.2) Write a Python program to implement Breadth First Search algorithm. Refer the following graph as an Input for the program.[Initial node=1,Goal node=8]. [20Marks



ANS:

```
graph={
    1:[2,3,4],
    2:[1,4,5],
    3:[1,4],
    4:[1,2,3,7],
    5:[2,6,7],
    6:[5,7],
    7:[4,5,6]
}
visited=[]
queue=[]
goal=7


def bfs(visited,graph,node):
   visited.append(node)
   queue.append(node)
   while queue:
      s=queue.pop(0)
```

```
        print(s,"->",end=" ")
        for neighbour in graph[s]:
            if neighbour not in visited:
                visited.append(neighbour)
                queue.append(neighbour)
                if goal in visited:
                    break
bfs(visited,graph,1)
```

OUTPUT:

```
1 -> 2 -> 3 -> 4 -> 5 -> 7 -> 6 ->
```

SLIP NO-07

Q.1)Write a python program implement tic-tac-toe using alpha beta pruning.

[10 Marks]

ANS:

```
def mytictactoe(val):
    print("\n")
    print("\t    |     |")
    print("\t {}  |  {}  |  {}".format(val[0], val[1], val[2]))
    print('\t_____|_____|_____')

    print("\t    |     |")
    print("\t {}  |  {}  |  {}".format(val[3], val[4], val[5]))
    print('\t_____|_____|_____')

    print("\t    |     |")
    print("\t {}  |  {}  |  {}".format(val[6], val[7], val[8]))
    print("\t    |     |")
    print("\n")

def myscoreboard(scoreboard):
    print("\t--------------------------------")
    print("\t              SCORE BOARD             ")
    print("\t--------------------------------")

    listofplayers = list(scoreboard.keys())
    print("\t  ", listofplayers[0], "\t    ", scoreboard[listofplayers[0]])
    print("\t  ", listofplayers[1], "\t    ", scoreboard[listofplayers[1]])
    print("\t--------------------------------\n")

def check_Victory(playerpos, curplayer):
    solution = [[1, 2, 3], [4, 5, 6], [7, 8, 9],
                [1, 4, 7], [2, 5, 8], [3, 6, 9],
                [1, 5, 9], [3, 5, 7]]

    for i in solution:
        if all(j in playerpos[curplayer] for j in i):
            return True
    return False

def check_Tie(playerpos):
    if len(playerpos['X']) + len(playerpos['O']) == 9:
        return True
    return False
```

```
def singlegame(curplayer, playerchoice):
    val = [' ' for i in range(9)]
    playerpos = {'X': [], 'O': []}

    while True:
        mytictactoe(val)

        try:
            print("Player", playerchoice[curplayer], "turn. Choose your block (1-9): ", end="")
            chance = int(input())
        except ValueError:
            print("Invalid Input !!! Try Again")
            continue

        if chance < 1 or chance > 9:
            print("Invalid Input !!! Try Again")
            continue

        if val[chance - 1] != ' ':
            print("Oops! The place is already occupied. Try again!")
            continue
        val[chance - 1] = curplayer
        playerpos[curplayer].append(chance)
        if check_Victory(playerpos, curplayer):
            mytictactoe(val)
            print("Congratulations! Player", playerchoice[curplayer], "has won the game!!\n")
            return curplayer

        if check_Tie(playerpos):
            mytictactoe(val)
            print("Game Tied\n")
            return 'D'

        curplayer = 'O' if curplayer == 'X' else 'X'

if __name__ == "__main__":
    print("First Player")
    FirstPlayer = input("Specify the Name: ")
    print("\n")

    print("Second Player")
    SecondPlayer = input("Specify the Name: ")
    print("\n")

    curplayer = FirstPlayer
    playerchoice = {'X': "", 'O': ""}
    scoreboard = {FirstPlayer: 0, SecondPlayer: 0}
    myscoreboard(scoreboard)

    while True:
        print(curplayer, "will make the choice:")
        print("Press 1 for X")
        print("Press 2 for O")
        print("Press 3 to Quit")

        try:
            the_choice = int(input())
        except ValueError:
            print("Invalid Input!!! Try Again\n")
            continue

        if the_choice == 1:
            playerchoice['X'] = curplayer
            playerchoice['O'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 2:
            playerchoice['O'] = curplayer
            playerchoice['X'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 3:
            print("The final scores")
            myscoreboard(scoreboard)
            break
        else:
            print("Invalid selection!! Try Again\n")
            continue


        win = singlegame('X' if the_choice == 1 else 'O', playerchoice)

        if win != 'D':
            playerWon = playerchoice[win]
            scoreboard[playerWon] = scoreboard[playerWon] + 1

        myscoreboard(scoreboard)
```

OUTPUT:

```
First Player
Specify the Name: sayali

Second Player
Specify the Name: sharii
```

```
          ----------------------------------
                    SCORE BOARD
          ----------------------------------
            sayali       0
            sharii       0
          ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
1


              |      |
              |      |
          ____|____ _|________
              |      |
              |      |
          ____|______|________
              |      |
              |      |
              |      |


Player sayali turn. Choose your block (1-9): 1


               |      |
           X   |      |
          _____|______|________
               |      |
               |      |
          _____|______|________
               |      |
               |      |
               |      |


Player sharii turn. Choose your block (1-9): 3
```

```
      |     |
   X  |     |  O
 _____|_____|_____
      |     |
      |     |
 _____|_____|_____
      |     |
      |     |
      |     |
```

Player sayali turn. Choose your block (1-9): 5

```
      |     |
   X  |     |  O
 _____|_____|_____
      |     |
      |  X  |
 ___  |_____|_____
      |     |
      |     |
      |     |
```

Player sharii turn. Choose your block (1-9): 9

```
      |    |
   X  |    | O
 _____|____|_____
      |    |
      |  X|
 _____|____|_____
      |    |
      |    | O
      |    |
```

Player sayali turn. Choose your block (1-9): 6

```
       |     |
  X    |     | O
  _____|_____|______
       |     |
       |  X  |  X
  _____|_____|______
       |     |
       |     | O
       |     |

Player sharii turn. Choose your block (1-9): 7

      |     |
  X   |     | O
  __  |__   |______
      |     |
      |  X  |  X
  ____|__   |______
      |     |
  O   |     | O
      |     |

Player sayali turn. Choose your block (1-9): 8

      |     |
  X   |     | O
  ____|__   |______
      |     |
      |  X  |  X
  ____|_____|______
      |     |
  O   |  X  | O
      |     |

Player sharii turn. Choose your block (1-9): 2
```

```
             |     |
          X  |  O  |  O
        _____|_____|_____
             |     |
             |  X  |  X
        _____|_____|_____
             |     |
          O  |  X  |  O
             |     |


Player sayali turn. Choose your block (1-9): 4


             |     |
          X  |  O  |  O
        _____|_____|_____
             |     |
          X  |  X  |  X
        _____|_____|_____
             |     |
          O  |  X  |  O
             |     |


Congratulations! Player sayali has won the game!!

        ----------------------------------
                   SCORE BOARD
        ----------------------------------
          sayali       1
          sharii       0
        ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
```

Q.2) Write a Python program to implement Simple Chatbot. [20Marks]

ANS:

```
from datetime import date

from datetime import datetime

import math

dict1 = {"date":"","time":"","sqr":"","fine":"I am also fine How can I help you","bye":"Bye Bye"}

print("Hi myself ABC chat my versions 1.0")

n=input("what is your name?=>")

print("Hi {} Hello! How are you?".format(n))

print("How can I call you?")

print("I can provide service")

for i in dict1.keys():

   print(i)

while(1):

   n=input("Enter your question")

   if n in dict1.keys():

      if(n=="date"):

         print("Today is=>",date.today())

      if(n=="time"):

         now=datetime.now()

         print("Current time is=>",now.strftime("%H:%M:%S"))

      if(n=="sqr"):

         m=int(input("Enter number=>"))

         print("sqr of {} is {}".format(m, m*m))
```

```
        if(n=="fine"):
            print(dict1[n])
        else:
            print("What you want to tell? Not getting your question")
    print("May I help you?")
    for k in dict1.keys():
        print(k)
```

OUTPUT:

```
Hi myself ABC chat my versions 1.0
what is your name?=>sayali
Hi sayali Hello! How are you?
How can I call you?
I can provide service
date
time
sqr
fine
bye
Enter your questiondate
Today is=> 2025-10-01
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questiontime
Current time is=> 11:07:56
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionsqr
Enter number=>4
```

```
sqr of 4 is 16
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionfine
I am also fine How can I help you
May I help you?
date
time
sqr
fine
bye
Enter your questionbye
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
```

SLIP NO-08

Q.1) Write a Python program to accept a string. Find and print the number of upper case alphabets and lower case alphabets. [10 Marks]

ANS:

```
n=input("Enter a String:")

upper_count=0

lower_count=0

for ch in n:

    if ch.isupper():

        upper_count+=1

    elif ch.islower():

        lower_count+=1

print("uppercase letter",upper_count)

print("uppercase letter",lower_count)
```

OUTPUT:

```
Enter a String:Hellow World
uppercase letter 2
uppercase letter 9
```

Q.2) Write a Python program to solve tic-tac-toe problem. [20 Marks]

ANS:

```
def mytictactoe(val):
    print("\n")
    print("\t     |     |")
    print("\t {}  |  {}  |  {}".format(val[0], val[1], val[2]))
    print('\t_____|_____|_____')

    print("\t     |     |")
    print("\t {}  |  {}  |  {}".format(val[3], val[4], val[5]))
    print('\t_____|_____|_____')

    print("\t     |     |")
    print("\t {}  |  {}  |  {}".format(val[6], val[7], val[8]))
    print("\t     |     |")
    print("\n")

def myscoreboard(scoreboard):
    print("\t--------------------------------")
    print("\t              SCORE BOARD       ")
    print("\t--------------------------------")

    listofplayers = list(scoreboard.keys())
    print("\t   ", listofplayers[0], "\t    ", scoreboard[listofplayers[0]])
    print("\t   ", listofplayers[1], "\t    ", scoreboard[listofplayers[1]])
    print("\t--------------------------------\n")

def check_Victory(playerpos, curplayer):
    solution = [[1, 2, 3], [4, 5, 6], [7, 8, 9],
                [1, 4, 7], [2, 5, 8], [3, 6, 9],
                [1, 5, 9], [3, 5, 7]]

    for i in solution:
        if all(j in playerpos[curplayer] for j in i):
            return True
    return False

def check_Tie(playerpos):
    if len(playerpos['X']) + len(playerpos['O']) == 9:
        return True
    return False

def singlegame(curplayer, playerchoice):
    val = [' ' for i in range(9)]
    playerpos = {'X': [], 'O': []}

    while True:
        mytictactoe(val)

        try:
            print("Player", playerchoice[curplayer], "turn. Choose your block (1-9): ", end="")
            chance = int(input())
        except ValueError:
            print("Invalid Input !!! Try Again")
            continue

        if chance < 1 or chance > 9:
            print("Invalid Input !!! Try Again")
            continue

        if val[chance - 1] != ' ':
            print("Oops! The place is already occupied. Try again!")
            continue
        val[chance - 1] = curplayer
        playerpos[curplayer].append(chance)
        if check_Victory(playerpos, curplayer):
            mytictactoe(val)
            print("Congratulations! Player", playerchoice[curplayer], "has won the game!!\n")
            return curplayer

        if check_Tie(playerpos):
            mytictactoe(val)
            print("Game Tied\n")
            return 'D'

        curplayer = 'O' if curplayer == 'X' else 'X'

if __name__ == "__main__":
    print("First Player")
    FirstPlayer = input("Specify the Name: ")
    print("\n")

    print("Second Player")
    SecondPlayer = input("Specify the Name: ")
    print("\n")

    curplayer = FirstPlayer
    playerchoice = {'X': "", 'O': ""}
    scoreboard = {FirstPlayer: 0, SecondPlayer: 0}
    myscoreboard(scoreboard)
    while True:
        print(curplayer, "will make the choice:")
        print("Press 1 for X")
        print("Press 2 for O")
        print("Press 3 to Quit")

        try:
            the_choice = int(input())
        except ValueError:
            print("Invalid Input!!! Try Again\n")
            continue

        if the_choice == 1:
            playerchoice['X'] = curplayer
            playerchoice['O'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 2:
            playerchoice['O'] = curplayer
            playerchoice['X'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 3:
            print("The final scores")
            myscoreboard(scoreboard)
            break
        else:
            print("Invalid selection!! Try Again\n")
            continue


        win = singlegame('X' if the_choice == 1 else 'O', playerchoice)

        if win != 'D':
            playerWon = playerchoice[win]
            scoreboard[playerWon] = scoreboard[playerWon] + 1

        myscoreboard(scoreboard)
```

OUTPUT:

```
First Player
Specify the Name: sayali

Second Player
Specify the Name: sharii
```

```
            ----------------------------------
                     SCORE BOARD
            ----------------------------------
              sayali        0
              sharii        0
            ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
1


               |      |
               |      |
            ___|___ __|_____
               |      |
               |      |
            ___|____ |_____
               |      |
               |      |
               |      |


Player sayali turn. Choose your block (1-9): 1


                |      |
             X  |      |
            ____|____ |_____
                |      |
                |      |
            ____|____ |_____
                |      |
                |      |
                |      |


Player sharii turn. Choose your block (1-9): 3


                |     |
             X  |     |  O
            ____|___  |_____
```

```
     |     |
     |     |
_____|___ |______
     |     |
     |     |
     |     |
```

Player sayali turn. Choose your block (1-9): 5

```
     |     |
  X  |     |  O
_____|___ |______
     |     |
     |  X |
___ |_____|______
     |     |
     |     |
     |     |
```

Player sharii turn. Choose your block (1-9): 9

```
     |    |
  X  |    | O
_____|___ |______
     |    |
     |  X|
_____|___ |______
     |    |
     |    | O
     |    |
```

Player sayali turn. Choose your block (1-9): 6

```
     |     |
  X  |     |  O
_____|_____|_____
     |     |
     |  X  |  X
_____|_____|_____
     |     |
     |     |  O
     |     |
```

```
Player sharii turn. Choose your block (1-9): 7
```

```
     |     |
  X  |     |  O
_____|_____|_____
     |     |
     |  X  |  X
_____|_____|_____
     |     |
  O  |     |  O
     |     |
```

```
Player sayali turn. Choose your block (1-9): 8
```

```
     |     |
  X  |     |  O
_____|_____|_____
     |     |
     |  X  |  X
_____|_____|_____
     |     |
  O  |  X  |  O
     |     |
```

```
Player sharii turn. Choose your block (1-9): 2
```

```
                  |       |
             X  |  O  |  O
           ___|___ |_____
                  |       |
                  |  X  |  X
           ___|___ |_____
                  |       |
             O  |  X  |  O
                  |       |


Player sayali turn. Choose your block (1-9): 4


                  |       |
             X  |  O  |  O
           ___|___ |_____
                  |       |
             X  |  X  |  X
           ___|___ |_____
                  |       |
             O  |  X  |  O
                  |       |


Congratulations! Player sayali has won the game!!

           ----------------------------------
                    SCORE BOARD
           ----------------------------------
             sayali       1
             sharii       0
           ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
```

SLIP NO-09

Q.1) Write python program to solve 8 puzzle problem using A* algorithm. [10 marks]

ANS:

```
import heapq

class PuzzleNode:

    def __init__(self, node_state, parent_node=None, move=None, cost=0):

        self.node_state = node_state

        self.parent_node = parent_node

        self.move = move

        self.cost = cost

        self.heuristic = self.calculate_heuristic()


    def __lt__(self, other):

        return (self.cost + self.heuristic) < (other.cost + other.heuristic)


    def calculate_heuristic(self):

        # Manhattan distance heuristic

        heuristic = 0

        for i in range(3):

            for j in range(3):

                if self.node_state[i][j] != 0:

                    goal_i, goal_j = divmod(self.node_state[i][j] - 1, 3)

                    heuristic += abs(i - goal_i) + abs(j - goal_j)

        return heuristic
```

```
def is_valid_move(i, j):
    return 0 <= i < 3 and 0 <= j < 3


def get_neighbors(node):
    neighbors = []
    i, j = next((i, j) for i in range(3) for j in range(3) if node.node_state[i][j] == 0)
    for di, dj in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        new_i, new_j = i + di, j + dj
        if is_valid_move(new_i, new_j):
            new_state = [row[:] for row in node.node_state]
            new_state[i][j], new_state[new_i][new_j] = new_state[new_i][new_j], new_state[i][j]
            neighbors.append(PuzzleNode(new_state, node, (i, j)))
    return neighbors


def reconstruct_path(node):
    path = []
    while node.parent_node is not None:
        path.append(node)
        node = node.parent_node
    return path[::-1]


def solve_8_puzzle(initial_state):
    start_node = PuzzleNode(initial_state)
    open_set = [start_node]
    closed_set = set()
    while open_set:
        current_node = heapq.heappop(open_set)
        if current_node.node_state == goal_state:
            return reconstruct_path(current_node)

        closed_set.add(tuple(map(tuple, current_node.node_state)))
        for neighbor in get_neighbors(current_node):
            if tuple(map(tuple, neighbor.node_state)) not in closed_set:
                heapq.heappush(open_set, neighbor)
    return None
if __name__ == "__main__":
    goal_state = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]  # Define the goal state
    initial_state = parse_input("__ed_input.txt")  # Parse initial state from input.txt
    path = solve_8_puzzle(initial_state)
    if path:
        print("Solution found! Moves to reach goal state:")
        for move in path:
            print(move.node_state)
    else:
        print("No solution found.")
```

Q.2) Write a Python program to solve water jug problem. 2 jugs with capacity 5 gallon and 7 gallon are given with unlimited water supply respectively. The target to achieve is 4 gallon of water in second jug. [15 Marks]

ANS:

```
def water_jug_dfs(capacity1,capacity2,target):
    visited=set()
    path=[]
    def dfs(jug1,jug2):
        if(jug1,jug2) in visited:
            return False
        visited.add((jug1,jug2))
        path.append((jug1,jug2))

        if jug1 == target or jug2 == target:
            return True

        if dfs(3,jug2):
            return True
        if dfs(jug1,5):
            return True
        if dfs(0,jug2):
            return True
        if dfs(jug1,0):
            return True

        if dfs(max(0,jug1-(5-jug2)),min(5,jug1+jug2)):
            return True

        if dfs(min(3,jug1+jug2),max(0,jug2-(3-jug1))):
            return True
        path.pop()
        return False

    dfs(0,0)
    return path

capacity1=5
capacity2=7
target=4

solution=water_jug_dfs(capacity1, capacity2,target)
if solution:
    print("Solution steps:")
    for step in solution:
        print(step)
else:
    print("No solution found.")
```

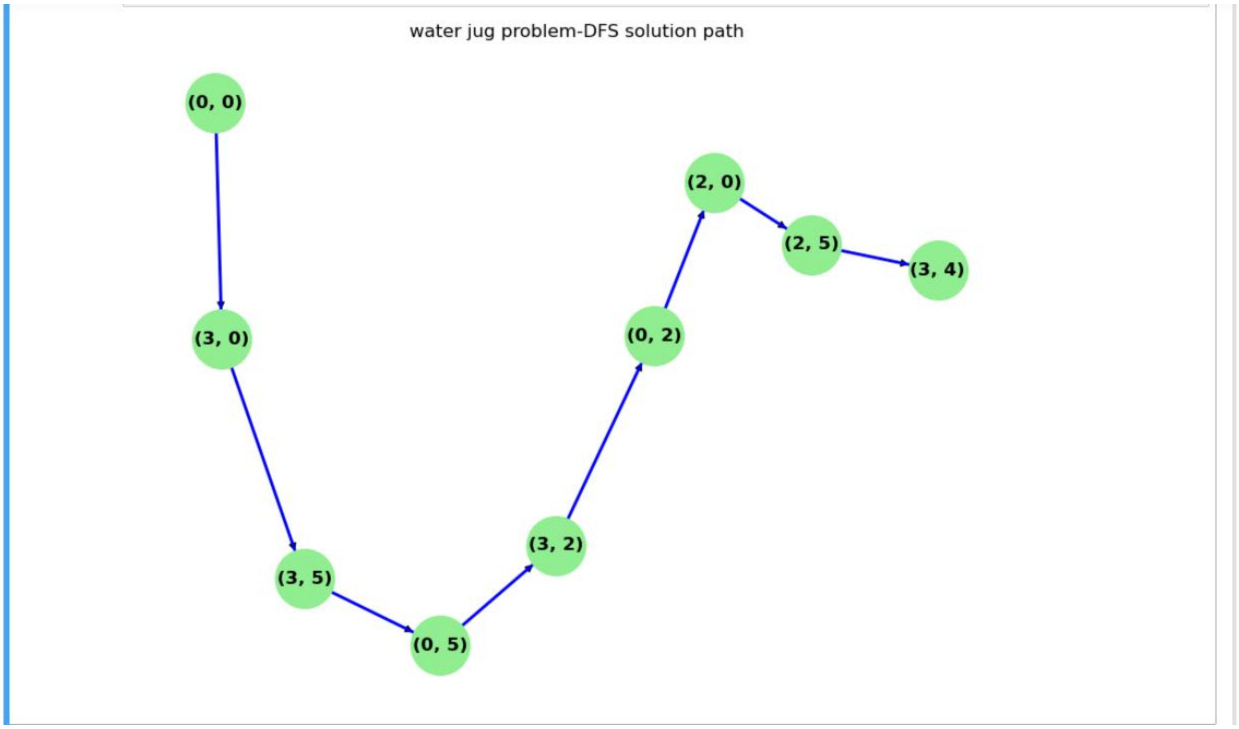
OUTPUT:

```
Solution steps:
(0, 0)
(3, 0)
```

```
(3, 5)
(0, 5)
(3, 2)
(0, 2)
(2, 0)
(2, 5)
(3, 4)
```

```
import matplotlib.pyplot as plt
import networkx as nx
def visualize_dfs_solution(solution):
    G=nx.DiGraph()
    for i in range(len(solution)-1):
        G.add_edge(solution[i],solution[i+1])


    pos=nx.spring_layout(G)
    plt.figure(figsize=(8,6))

nx.draw(G,pos,with_labels=True,node_color='lightgreen',node_size=1500,font_size=12,font_weight='bold')
    nx.draw_networkx_edges(G,pos,edgelist=list(G.edges()),edge_color='blue',width=2)
    plt.title("water jug problem-DFS solution path")
    plt.show()


if solution:
    visualize_dfs_solution(solution)
```

OUTPUT:



SLIP NO-10

Q.1) Write Python program to implement crypt arithmetic problem TWO+TWO=FOUR.

[10 Marks]

ANS:

```
import itertools
letters = ['T', 'W', 'O', 'F', 'U', 'R']


for perm in itertools.permutations(range(10), len(letters)):
    T, W, O, F, U, R = perm
    if T == 0 or F == 0:
        continue
    two = 100 * T + 10 * W + O
    four = 1000 * F + 100 * O + 10 * U + R
    if two + two == four:
        print(f"Solution found:")
        print(f"T={T}, W={W}, O={O}, F={F}, U={U}, R={R}")
        print(f"{two} + {two} = {four}")
        print()
```

OUTPUT:

```
Solution found:
T=7, W=3, O=4, F=1, U=6, R=8
734 + 734 = 1468

Solution found:
T=7, W=6, O=5, F=1, U=3, R=0
765 + 765 = 1530
```

```
Solution found:
T=8, W=3, O=6, F=1, U=7, R=2
836 + 836 = 1672

Solution found:
T=8, W=4, O=6, F=1, U=9, R=2
846 + 846 = 1692

Solution found:
T=8, W=6, O=7, F=1, U=3, R=4
867 + 867 = 1734

Solution found:
T=9, W=2, O=8, F=1, U=5, R=6
928 + 928 = 1856

Solution found:
T=9, W=3, O=8, F=1, U=7, R=6
938 + 938 = 1876
```

Q.2) Write a Python program to implement Simple Chatbot. [20 Marks]

ANS:

```
from datetime import date

from datetime import datetime

import math

dict1 = {"date":"","time":"","sqr":"","fine":"I am also fine How can I help you","bye":"Bye Bye"}

print("Hi myself ABC chat my versions 1.0")

n=input("what is your name?=>")

print("Hi {} Hello! How are you?".format(n))

print("How can I call you?")

print("I can provide service")

for i in dict1.keys():

    print(i)

while(1):

    n=input("Enter your question")

    if n in dict1.keys():

        if(n=="date"):

            print("Today is=>",date.today())

        if(n=="time"):

            now=datetime.now()

            print("Current time is=>",now.strftime("%H:%M:%S"))

        if(n=="sqr"):

            m=int(input("Enter number=>"))

            print("sqr of {} is {}".format(m, m*m))
        if(n=="fine"):
            print(dict1[n])
        else:
            print("What you want to tell? Not getting your question")
    print("May I help you?")
    for k in dict1.keys():
        print(k)
```

OUTPUT:

```
Hi myself ABC chat my versions 1.0
what is your name?=>sayali
Hi sayali Hello! How are you?
How can I call you?
I can provide service
date
time
sqr
fine
bye
Enter your questiondate
Today is=> 2025-10-01
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questiontime
Current time is=> 11:07:56
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionsqr
Enter number=>4
```

```
sqr of 4 is 16
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionfine
I am also fine How can I help you
May I help you?
date
time
sqr
fine
bye
Enter your questionbye
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
```

SLIP NO-11

Q.1) Write a python program using mean end analysis algorithm problem of transforming a string of lowercase letters into another string. [10 Marks]

ANS:

```
def mean_end_analysis(start, goal):
    current = list(start)
    target = list(goal)
    steps = []


    i = 0
    while i < len(current) or i < len(target):
        if i < len(current) and i < len(target):
            if current[i] != target[i]:
                steps.append(f"Replace '{current[i]}' with '{target[i]}' at position {i}")
                current[i] = target[i]
        elif i < len(current):
            steps.append(f"Delete '{current[i]}' at position {i}")
            current.pop(i)
            i -= 1  # Stay at same index after delete
        elif i < len(target):
            steps.append(f"Insert '{target[i]}' at position {i}")
            current.insert(i, target[i])
        i += 1


    return steps


# === Example ===start = "cat"
```

```
goal = "cart"
steps = mean_end_analysis(start, goal)
print(f"Transform '{start}' to '{goal}':\n")
for step in steps:
    print(step)
```

OUTPUT:

```
Transforming 'cat' to 'dog' using Means-End Analysis:


→ Replace 'c' with 'd' at position 0
→ Replace 'a' with 'o' at position 1
→ Replace 't' with 'g' at position 2


Final result: 'dog'
```

Q.2) Write a Python program to solve water jug problem. Two jugs with capacity 4 gallon and 3 gallon are given with unlimited water supply respectively. The target is to achieve 2 gallon of water in second jug. [10 Marks]

ANS:

```
def water_jug_dfs(capacity1,capacity2,target):
    visited=set()
    path=[]
    def dfs(jug1,jug2):
        if(jug1,jug2) in visited:
            return False
        visited.add((jug1,jug2))
        path.append((jug1,jug2))

        if jug1 == target or jug2 == target:
            return True

        if dfs(3,jug2):
            return True
        if dfs(jug1,5):
            return True
        if dfs(0,jug2):
            return True
        if dfs(jug1,0):
            return True

        if dfs(max(0,jug1-(5-jug2)),min(5,jug1+jug2)):
            return True

        if dfs(min(3,jug1+jug2),max(0,jug2-(3-jug1))):
            return True

        path.pop()
        return False

    dfs(0,0)
    return path

capacity1=4
capacity2=3
target=2

solution=water_jug_dfs(capacity1, capacity2,target)
if solution:
    print("Solution steps:")
    for step in solution:
        print(step)
else:
    print("No solution found.")
```

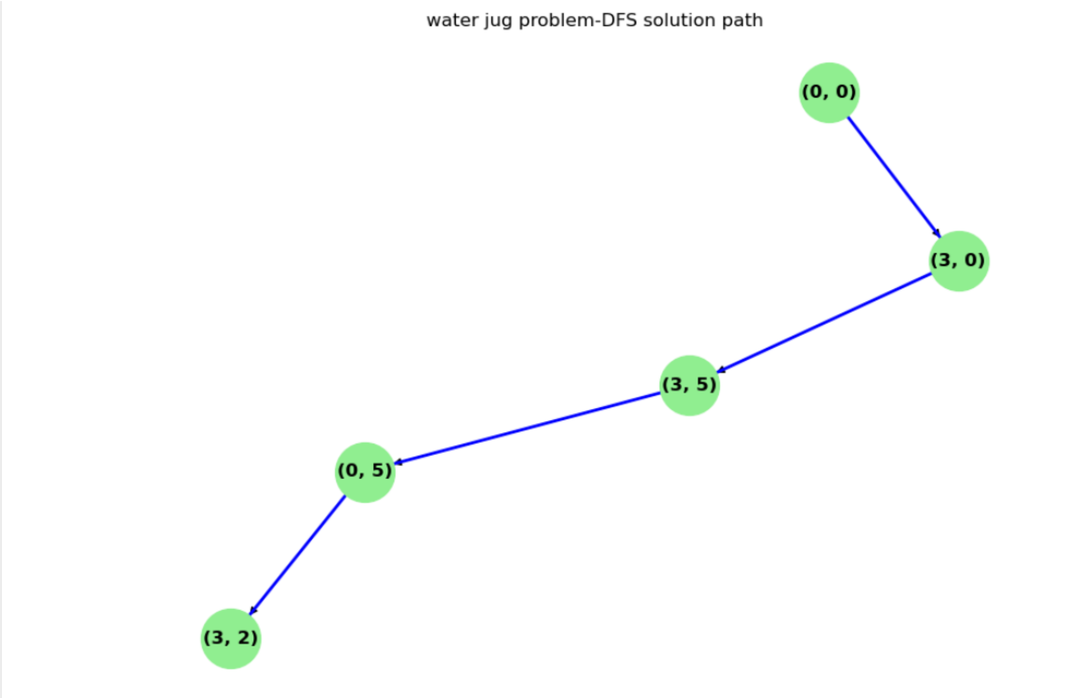
OUTPUT:

```
Solution steps:
```

```
(0, 0)
(3, 0)
(3, 5)
(0, 5)
(3, 2)
```

```
import matplotlib.pyplot as plt
import networkx as nx

def visualize_dfs_solution(solution):
    G=nx.DiGraph()
    for i in range(len(solution)-1):
        G.add_edge(solution[i],solution[i+1])

    pos=nx.spring_layout(G)
    plt.figure(figsize=(8,6))
    nx.draw(G,pos,with_labels=True,node_color='lightgreen',node_size=1500,font_size=12,font_weight='bold')
    nx.draw_networkx_edges(G,pos,edgelist=list(G.edges()),edge_color='blue',width=2)
    plt.title("water jug problem-DFS solution path")
    plt.show()

if solution:
    visualize_dfs_solution(solution)
```

OUTPUT:



SLIP NO-12

Q.1) Write a python program to generate Calendar for the given month and year?.

[10Marks]

ANS:

```
import calendar

from datetime import datetime


current=datetime.now()

print(calendar.month(current.year, current.month))
```

OUTPUT:

```
    October 2025
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Q.2)Write a Python program to simulate 4-Queens problem. [20 Marks]

ANS:

```
global N

N = 4

def printSolution(board):

    for i in range(N):

        for j in range(N):

            print (board[i][j],end=' ')

        print()

def isSafe(board, row, col):

    for i in range(col):

        if board[row][i] == 1:

            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    return True

def solveNQUtil(board, col):

    if col >= N:

        return True

    for i in range(N):

        if isSafe(board, i, col):
            board[i][col] = 1

            if solveNQUtil(board, col + 1) == True:

                return True

            board[i][col] = 0

    return False

def solveNQ():

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

              [0, 0, 0, 0],

              [0, 0, 0, 0],

              [0, 0, 0, 0]

             ]

    if solveNQUtil(board, 0) == False:

        print ("Solution does not exist")

        return False

    printSolution(board)

    return True

solveNQ()
```

OUTPUT:

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
True
```

SLIP NO-13

Q.1Write a Python program to implement Mini-Max Algorithm. [10 Marks]

ANS:

```
import math
def is_moves_left(board):
    for row in board:
        if "_" in row:
            return True
    return False
def evaluate(board):
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != "_":
            return 10 if row[0] == "X" else -10
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and board[0][col] != "_":
            return 10 if board[0][col] == "X" else -10
    if board[0][0] == board[1][1] == board[2][2] and board[0][0] != "_":
        return 10 if board[0][0] == "X" else -10
    if board[0][2] == board[1][1] == board[2][0] and board[0][2] != "_":
        return 10 if board[0][2] == "X" else -10
    return 0
def minimax(board, depth, is_max):
    score = evaluate(board)
    if score == 10:
        return score - depth
    if score == -10:
        return score + depth
    if not is_moves_left(board):
        return 0
    if is_max:
        best = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == "_":
                    board[i][j] = "X"
                    best = max(best, minimax(board, depth + 1, False))
                    board[i][j] = "_"
        return best
    else:
        best = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == "_":
                    board[i][j] = "O"
                    best = min(best, minimax(board, depth + 1, True))
                    board[i][j] = "_"
        return best
def find_best_move(board):
    best_val = -math.inf
    best_move = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == "_":
                board[i][j] = "X"
                move_val = minimax(board, 0, False)
                board[i][j] = "_"

                if move_val > best_val:
                    best_val = move_val
                    best_move = (i, j)

    return best_move
board = [
    ["X", "O", "X"],
    ["O", "O", "_"],
    ["_", "_", "_"]
]
best = find_best_move(board)
print("Best Move for X is:", best)
```

OUTPUT:

```
Best Move for X is: (1, 2)
```

Q.2) Write a Python program to simulate 8-Queens problem. [20 Marks]

ANS:

```
global N

N = 8

def printSolution(board):

    for i in range(N):

        for j in range(N):

            print (board[i][j],end=' ')

        print()

def isSafe(board, row, col):

    for i in range(col):

        if board[row][i] == 1:

            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    return True

def solveNQUtil(board, col):

    if col >= N:

        return True

    for i in range(N):

        if isSafe(board, i, col):
            board[i][col] = 1
            if solveNQUtil(board, col + 1) == True:
                return True
            board[i][col] = 0
    return False
def solveNQ():
    board = [ [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0]
             ]
    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False
    printSolution(board)
    return True
solveNQ()
```

OUTPUT:

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
True
```

SLIP NO-14

Q.1) Write a python program to sort the sentence in alphabetical order? [10Marks]

ANS:

```
my_str="Hello this Is an Example With cased letters"

words=[word.lower() for word in my_str.split()]

words.sort()

print("The sorted words are:")

for word in words:

    print(word)
```

OUTPUT:

```
The sorted words are:
an
cased
example
hello
is
letters
this
with
```

Q.2) Write a Python program to simulate n-Queens problem. [20Marks]

ANS:

```
global N

N = 4

def printSolution(board):

    for i in range(N):

        for j in range(N):

            print (board[i][j],end=' ')

        print()

def isSafe(board, row, col):

    for i in range(col):

        if board[row][i] == 1:

            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):

        if board[i][j] == 1:

            return False

    return True

def solveNQUtil(board, col):

    if col >= N:

        return True

    for i in range(N):

        if isSafe(board, i, col):

            board[i][col] = 1
            if solveNQUtil(board, col + 1) == True:
                return True
            board[i][col] = 0
    return False
def solveNQ():
    board = [ [0, 0, 0, 0],
              [0, 0, 0, 0],
              [0, 0, 0, 0],
              [0, 0, 0, 0]
             ]
    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False
    printSolution(board)
    return True
solveNQ()
```

OUTPUT:

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
True
```

SLIP NO-15

Q.1)Write a Program to Implement Monkey Banana Problem using Python.

[10 Marks]

ANS:

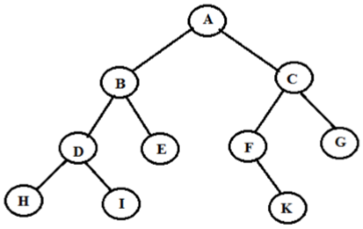
```
from collections import deque
def get_successors(state):
    monkey_pos, box_pos, on_box, has_banana = state
    successors = []
    if has_banana:
        return successors  # goal reached
    positions = ["door", "window", "middle"]
    for pos in positions:
        if pos != monkey_pos:
            successors.append((pos, box_pos, False, has_banana))
    if monkey_pos == box_pos and not on_box:
        for pos in positions:
            if pos != box_pos:
                successors.append((pos, pos, False, has_banana))
    if monkey_pos == box_pos and not on_box:
        successors.append((monkey_pos, box_pos, True, has_banana))
    if on_box and monkey_pos == "middle":
        successors.append((monkey_pos, box_pos, True, True))
    return successors
def bfs(start_state):
    visited = set()
    queue = deque([(start_state, [])])
    while queue:
        state, path = queue.popleft()
        if state in visited:
            continue
        visited.add(state)
        monkey_pos, box_pos, on_box, has_banana = state
        if has_banana:
            return path + [state]
        for next_state in get_successors(state):
            queue.append((next_state, path + [state]))
    return None
start = ("door", "window", False, False)
solution = bfs(start)
print("\nMonkey–Banana Solution Path:\n")
for step in solution:
    print(step)
```

OUTPUT:

```
('door', 'window', False, False)
('window', 'window', False, False)
('middle', 'middle', False, False)
('middle', 'middle', True, False
('middle', 'middle', True, True)
```

Q.2) Write a program to implement Iterative Deepening DFS algorithm. [20 Marks]

[Goal Node =G]



ANS:

```
graph = {
 'A' : ['B','C'],
 'B' : ['D', 'E'],
 'C' : ['F', 'G'],
 'D' : [‘H’,’I’],
 'E' : [],
 'F' : [K],
 'G' : []
}
path = list()
def DFS(currentNode,destination,graph,maxDepth,curList):
    curList.append(currentNode)
    if currentNode==destination:
        return True
    if maxDepth<=0:
        path.append(curList)
        return False
    for node in graph[currentNode]:
        if DFS(node,destination,graph,maxDepth-1,curList):
            return True
        else:
            curList.pop()
    return False
def iterativeDDFS(currentNode,destination,graph,maxDepth):
    for i in range(maxDepth):
        curList = list()
        if DFS(currentNode,destination,graph,i,curList):
            return True
    return False
if not iterativeDDFS('A','G',graph,3):
    print("Path is not available")
else:
    print("Path exists")
    print(path.pop())
```

SLIP NO -16

Q.1) Write a Program to Implement Tower of Hanoi using Python. [10 Marks]

ANS:

```
def tower_of_hanoi(n, source, auxiliary, destination):
    if n == 1:
        print(f"Move disk 1 from {source} to {destination}")
        return
    tower_of_hanoi(n - 1, source, destination, auxiliary)
    print(f"Move disk {n} from {source} to {destination}")
    tower_of_hanoi(n - 1, auxiliary, source, destination)


n = int(input("Enter number of disks: "))
print("\nSteps to solve Tower of Hanoi:\n")
tower_of_hanoi(n, 'A', 'B', 'C')
```

OUTPUT:

```
Enter number of disks: 2

Steps to solve Tower of Hanoi:

Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
```

Q.2) Write a Python program to solve tic-tac-toe problem. [15 Marks]

ANS:

```
def mytictactoe(val):
    print("\n")
    print("\t    |    |")
    print("\t {}  |  {}  |  {}".format(val[0], val[1], val[2]))
    print('\t_____|_____|_____')

    print("\t    |    |")
    print("\t {}  |  {}  |  {}".format(val[3], val[4], val[5]))
    print('\t_____|_____|_____')

    print("\t    |    |")
    print("\t {}  |  {}  |  {}".format(val[6], val[7], val[8]))
    print("\t    |    |")
    print("\n")

def myscoreboard(scoreboard):
    print("\t--------------------------------")
    print("\t              SCORE BOARD              ")
    print("\t--------------------------------")

    listofplayers = list(scoreboard.keys())
    print("\t  ", listofplayers[0], "\t    ", scoreboard[listofplayers[0]])
    print("\t  ", listofplayers[1], "\t    ", scoreboard[listofplayers[1]])
    print("\t--------------------------------\n")

def check_Victory(playerpos, curplayer):
    solution = [[1, 2, 3], [4, 5, 6], [7, 8, 9],
                [1, 4, 7], [2, 5, 8], [3, 6, 9],
                [1, 5, 9], [3, 5, 7]]

    for i in solution:
        if all(j in playerpos[curplayer] for j in i):
            return True
    return False

def check_Tie(playerpos):
    if len(playerpos['X']) + len(playerpos['O']) == 9:
        return True
    return False

def singlegame(curplayer, playerchoice):
    val = [' ' for i in range(9)]
    playerpos = {'X': [], 'O': []}
    while True:
        mytictactoe(val)

        try:
            print("Player", playerchoice[curplayer], "turn. Choose your block (1-9): ", end="")
            chance = int(input())
        except ValueError:
            print("Invalid Input !!! Try Again")
            continue

        if chance < 1 or chance > 9:
            print("Invalid Input !!! Try Again")
            continue

        if val[chance - 1] != ' ':
            print("Oops! The place is already occupied. Try again!")
            continue
        val[chance - 1] = curplayer
        playerpos[curplayer].append(chance)
        if check_Victory(playerpos, curplayer):
            mytictactoe(val)
            print("Congratulations! Player", playerchoice[curplayer], "has won the game!!\n")
            return curplayer

        if check_Tie(playerpos):
            mytictactoe(val)
            print("Game Tied\n")
            return 'D'

        curplayer = 'O' if curplayer == 'X' else 'X'

if __name__ == "__main__":
    print("First Player")
    FirstPlayer = input("Specify the Name: ")
    print("\n")

    print("Second Player")
    SecondPlayer = input("Specify the Name: ")
    print("\n")

    curplayer = FirstPlayer
    playerchoice = {'X': "", 'O': ""}
    scoreboard = {FirstPlayer: 0, SecondPlayer: 0}
    myscoreboard(scoreboard)
    while True:
        print(curplayer, "will make the choice:")
        print("Press 1 for X")
        print("Press 2 for O")
        print("Press 3 to Quit")

        try:
            the_choice = int(input())
        except ValueError:
            print("Invalid Input!!! Try Again\n")
            continue

        if the_choice == 1:
            playerchoice['X'] = curplayer
            playerchoice['O'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 2:
            playerchoice['O'] = curplayer
            playerchoice['X'] = SecondPlayer if curplayer == FirstPlayer else FirstPlayer
        elif the_choice == 3:
            print("The final scores")
            myscoreboard(scoreboard)
            break
        else:
            print("Invalid selection!! Try Again\n")
            continue


        win = singlegame('X' if the_choice == 1 else 'O', playerchoice)

        if win != 'D':
            playerWon = playerchoice[win]
            scoreboard[playerWon] = scoreboard[playerWon] + 1
+
        myscoreboard(scoreboard)
```

OUTPUT:

```
First Player
Specify the Name: sayali

Second Player
Specify the Name: sharii
```

```
        ----------------------------------
                 SCORE BOARD
        ----------------------------------
          sayali       0
          sharii       0
        ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
1


           |      |
           |      |
        ___|__ _|______
           |      |
           |      |
        ___|____ |______
           |      |
           |      |
           |      |


Player sayali turn. Choose your block (1-9): 1


             |      |
         X   |      |
        _____|____ |______
             |      |
             |      |
        _____|____ |______
             |      |
             |      |
             |      |


Player sharii turn. Choose your block (1-9): 3
```

```
     |     |
  X  |     |  O
_____|___  |_____
     |     |
     |     |
_____|___  |_____
     |     |
     |     |
     |     |

Player sayali turn. Choose your block (1-9): 5

     |     |
  X  |     |  O
_____|___  |_____
     |     |
     |  X  |
___  |_____|_____
     |     |
     |     |
     |     |

Player sharii turn. Choose your block (1-9): 9

     |    |
  X  |    |  O
_____|___ |_____
     |    |
     |  X|
_____|___ |_____
     |    |
     |    |  O
     |    |

Player sayali turn. Choose your block (1-9): 6
```

```
     |     |
 X   |     | O
_____|_____|_____
     |     |
     |  X  |  X
_____|_____|_____
     |     |
     |     | O
     |     |
```

Player sharii turn. Choose your block (1-9): 7

```
     |     |
 X   |     | O
____ |____ |_____
     |     |
     |  X  |  X
_____|____ |_____
     |     |
 O   |     | O
     |     |
```

Player sayali turn. Choose your block (1-9): 8

```
     |     |
 X   |     | O
_____|____ |_____
     |     |
     |  X  |  X
_____|_____|_____
     |     |
 O   |  X  | O
     |     |
```

Player sharii turn. Choose your block (1-9): 2

```
                     |     |
                  X  |  O  |  O
                _____|_____|_____
                     |     |
                     |  X  |  X
                _____|_____|_____
                     |     |
                  O  |  X  |  O
                     |     |


Player sayali turn. Choose your block (1-9): 4


                     |     |
                  X  |  O  |  O
                _____|_____|_____
                     |     |
                  X  |  X  |  X
                _____|_____|_____
                     |     |
                  O  |  X  |  O
                     |     |


Congratulations! Player sayali has won the game!!

                ----------------------------------
                           SCORE BOARD
                ----------------------------------
                  sayali       1
                  sharii       0
                ----------------------------------

sayali will make the choice:
Press 1 for X
Press 2 for O
Press 3 to Quit
```

SLIP NO-17

Q.1) Python program that demonstrates the hill climbing algorithm to find the maximum of a mathematical function. [10Marks]

ANS:

```
import numpy as np

def objective_function(x):

    return -x**2+4*x

def hill_climbing(start_x,step_size,num_steps):

    current_x=start_x

    for step in range(num_steps):

        current_value=objective_function(current_x)

        next_x=current_x+step_size

        next_value=objective_function(next_x)

        if next_value > current_value:

            current_x=next_x

        return current_x,objective_function(current_x)

start_x=0.0

step_size=0.1

num_steps=100

best_x,best_value=hill_climbing(start_x,step_size,num_steps)

print(f"Starting point:{start_x:.2f}")

print(f"optimal x:{best_x:.2f}")

print(f"Minimum value:{best_value:.2f}")
```

```
OUTPUT:         Starting point:0.00

                optimal x:0.10
               Minimum value:0.39
```

SLIP NO-18

Q.1).Write a python program to remove stop words for a given passage from a text file using NLTK?. [10Marks]

ANS:

```
import nltk

from nltk.corpus import stopwords

with open("input.txt", "r") as file:

    text = file.read()

stop_words = set(stopwords.words("english"))

words = text.split()

filtered_words = [word for word in words if word.lower() not in stop_words]

clean_text = " ".join(filtered_words)

with open("output.txt", "w") as file:

    file.write(clean_text)

print("Stop words removed successfully! Check output.txt")
```

OUTPUT:

Input.txt

```
This is a sample passage demonstrating the removal of stop words using NLTK in Python.
```

Output.txt

```
sample passage demonstrating removal stop words using NLTK Python.
```

SLIP NO-19

Q.1) Write a program to implement Hangman game using python.

Description: Hangman is a classic word-guessing game. The user should guess the word correctly by entering alphabets of the user choice. The Program will get input as single alphabet from the user and it will matchmaking with the alphabets in the original word.

[10 Marks]

ANS:

```
import time
import random
name=input("what is your name?")
print("Hello,"+name,"Time to play hangman!")
time.sleep(1)
print("Start guessing...\n")


time.sleep(0.5)
words=['python','programming','treasure','creative','medium','horror']
word=random.choice(words)
guesses=""
turns=10
while turns>0:
    failed=0
    for char in words:
        if char in guesses:
            print(char,end="")
        else:
            print("*",end=""),
            failed+=1
    if failed==0:
        print("\n You Won")
        break
    guess=input("\nguess the character:")
    guesses+=guess
    if guess not in word:
        turns-=1
        print("\nWrong")
        print("\nYou have",+turns,'more guesses')
        if turns==0:
            print("\n You Loss")
```

OUTPUT:

```
what is your name?mcs
Hello,mcs Time to play hangman!
Start guessing...


******
guess the character:p
******
guess the character:y


Wrong
```

```
You have 9 more guesses
******
guess the character:t

Wrong


You have 8 more guesses
******
guess the character:h

Wrong


You have 7 more guesses
******
guess the character:o
******
guess the character:n
python*****
```

SLIP NO -20

Q.1) Build a bot which provides all the information related to you in college.

[10Marks]

ANS:

```
from datetime import date, datetime

college_info = {

    "principal": "Dr.Mahamuni sir.",

    "courses": "We offer BSc Computer Science, BCom, BA, MSc, and other PG courses.",

    "departments": "We have departments like Computer Science, Commerce, Arts, Physics, Chemistry, and Mathematics.",

    "library": "Our library has more than 20,000 books, e-journals, and a digital study zone.",

    "events": "Annual Day, Tech Fest, Cultural Programs, and Sports Week are conducted every year.",

    "labs": "We have Computer Labs, Electronics Labs, and a Language Lab.",

    "canteen": "The college canteen provides hygienic and affordable food.",

    "date": f"Today is {date.today()}",

    "time": f"Current time is {datetime.now().strftime('%H:%M:%S')}",

    "bye": "Bye Bye! Have a great day!"

}


print("Hi, I am *College Info Bot v1.0*")

name = input("What is your name? => ")

print("Hi {}, Welcome to our college chatbot!".format(name))

print("You can ask me about: ")

for i in college_info.keys():

    print("-", i)


while True:
```

```
query = input("\nEnter your question: ").lower()
if query in college_info:
    print(college_info[query])
    if query == "bye":
        break
else:
    print("Sorry, I don't have information about that. Please ask something else.")


print("\nYou can ask me about:", ", ".join(college_info.keys()))
```

Q.2) Write a Python program to implement Mini-Max Algorithm. [20Marks]

ANS:

```
import math
def is_moves_left(board):
    for row in board:
        if "_" in row:
            return True
    return False
def evaluate(board):
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != "_":
            return 10 if row[0] == "X" else -10
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and board[0][col] != "_":
            return 10 if board[0][col] == "X" else -10
    if board[0][0] == board[1][1] == board[2][2] and board[0][0] != "_":
        return 10 if board[0][0] == "X" else -10
    if board[0][2] == board[1][1] == board[2][0] and board[0][2] != "_":
        return 10 if board[0][2] == "X" else -10
    return 0
def minimax(board, depth, is_max):
    score = evaluate(board)
    if score == 10:
        return score - depth


    if score == -10:
        return score + depth
    if not is_moves_left(board):
        return 0
    if is_max:
        best = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == "_":
                    board[i][j] = "X"
                    best = max(best, minimax(board, depth + 1, False))
                    board[i][j] = "_"
        return best
    else:
        best = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == "_":
                    board[i][j] = "O"
                    best = min(best, minimax(board, depth + 1, True))
                    board[i][j] = "_"
        return best
def find_best_move(board):
    best_val = -math.inf
    best_move = (-1, -1)

    for i in range(3):
        for j in range(3):
            if board[i][j] == "_":
                board[i][j] = "X"
                move_val = minimax(board, 0, False)
                board[i][j] = "_"

                if move_val > best_val:
                    best_val = move_val
                    best_move = (i, j)

    return best_move
board = [
    ["X", "O", "X"],
    ["O", "O", "_"],
    ["_", "_", "_"]
]
best = find_best_move(board)
print("Best Move for X is:", best)
```

OUTPUT:

```
Best Move for X is: (1, 2)
```

SLIP NO-21

Q.1)Write a python program to remove punctuations from the given string? [10 Marks]

ANS:

```
str=input("Enter a string:")
for c in str:
    if c in "@'!":
        str=str.replace(c,")
print(str)
```

```
OUTPUT:Enter a string:ADT's!college
ADTscollege
```

Q.2)Write a Python program for the following Cryptarithmetic problems. [20 Marks]

GO + TO = OUT

ANS:

```
import itertools

letters = ['T',  'O', 'G', 'U']

for perm in itertools.permutations(range(10), len(letters)):

    T, O, G, U  = perm

    if T == 0 or O == 0 or G==0:

        continue

    to = 10 * T + O

    go = 10 * G + O

    out=100*O+10*U+T

    if to+go==out:

        print(f"Solution found:")

        print(f"T={T},O={O}, U={U}, G={G}")

        print(f"{to} + {go} = {out}")

        print()
```

OUTPUT:

```
Solution found:
T=2,O=1, U=0, G=8
21 + 81 = 102
```

SLIP NO-22

Q.1) Write a Program to Implement Alpha-Beta Pruning using Python. [10 Marks]

ANS:

```
import math
def alpha_beta_pruning(depth, node_index, is_maximizing_player, values, alpha, beta, max_depth):
    if depth == max_depth:
        return values[node_index]

    if is_maximizing_player:
        best = -math.inf
        # Explore left and right child nodes
        for i in range(2):
            val = alpha_beta_pruning(depth + 1, node_index * 2 + i, False, values, alpha, beta, max_depth)
            best = max(best, val)
            alpha = max(alpha, best)
            # Prune branch
            if beta <= alpha:
                break
        return best
    else:
        best = math.inf
        # Explore left and right child nodes
        for i in range(2):
            val = alpha_beta_pruning(depth + 1, node_index * 2 + i, True, values, alpha, beta, max_depth)
            best = min(best, val)
            beta = min(beta, best)
            # Prune branch
            if beta <= alpha:
                break
        return best
values = [3, 5, 6, 9, 1, 2, 0, -1]
max_depth = 3
print("Leaf node values:", values)
optimal_value = alpha_beta_pruning(0, 0, True, values, -math.inf, math.inf, max_depth)
print("Optimal value (after Alpha-Beta Pruning):", optimal_value)
```

OUTPUT:

```
Leaf node values: [3, 5, 6, 9, 1, 2, 0, -1]
Optimal value (after Alpha-Beta Pruning): 5
```

Q.2) Write a Python program to implement Simple Chatbot. [20 Marks]

ANS:

```
from datetime import date

from datetime import datetime

import math

dict1 = {"date":"","time":"","sqr":"","fine":"I am also fine How can I help you","bye":"Bye Bye"}

print("Hi myself ABC chat my versions 1.0")

n=input("what is your name?=>")

print("Hi {} Hello! How are you?".format(n))

print("How can I call you?")

print("I can provide service")

for i in dict1.keys():

    print(i)

while(1):

    n=input("Enter your question")

    if n in dict1.keys():

        if(n=="date"):

            print("Today is=>",date.today())

        if(n=="time"):

            now=datetime.now()

            print("Current time is=>",now.strftime("%H:%M:%S"))

        if(n=="sqr"):

            m=int(input("Enter number=>"))

            print("sqr of {} is {}".format(m, m*m))
        if(n=="fine"):

            print(dict1[n])

        else:

            print("What you want to tell? Not getting your question")

    print("May I help you?")

    for k in dict1.keys():

        print(k)
```

OUTPUT:

```
Hi myself ABC chat my versions 1.0
what is your name?=>sayali
Hi sayali Hello! How are you?
How can I call you?
I can provide service
date
time
sqr
fine
bye
Enter your questiondate
Today is=> 2025-10-01
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questiontime
Current time is=> 11:07:56
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionsqr
Enter number=>4
```

```
sqr of 4 is 16
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
Enter your questionfine
I am also fine How can I help you
May I help you?
date
time
sqr
fine
bye
Enter your questionbye
What you want to tell? Not getting your question
May I help you?
date
time
sqr
fine
bye
```

SLIP NO- 23

Q.1) Write a Program to Implement Tower of Hanoi using Python. [10 Marks]

ANS:

```
def tower_of_hanoi(n, source, auxiliary, destination):
    if n == 1:
        print(f"Move disk 1 from {source} to {destination}")
        return
    tower_of_hanoi(n - 1, source, destination, auxiliary)
    print(f"Move disk {n} from {source} to {destination}")
    tower_of_hanoi(n - 1, auxiliary, source, destination)


n = int(input("Enter number of disks: "))
print("\nSteps to solve Tower of Hanoi:\n")
tower_of_hanoi(n, 'A', 'B', 'C')
```

OUTPUT:

```
Enter number of disks: 2

Steps to solve Tower of Hanoi:

Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
```

Q.2) Write a Python program for the following Cryptarithmetic problems SEND + MORE = MONEY [20Marks]

ANS:

```
import itertools

letters = ['S','E', 'N', 'D','M','O','R','Y']

for perm in itertools.permutations(range(10), len(letters)):

   S,E,N,D,M,O,R,Y  = perm

   if S == 0 or M == 0:

      continue

   send = 1000*S+100*E+10 * N + D

   more = 1000*M+100*O+10 * R + E

   money=10000*M+1000*O+100*N+10*E+Y

   if send+more==money:

      print(f"Solution found:")

      print(f"S={S},E={E},N={N},D={D},M={M},O={O},R={R},Y={Y}")

      print(f"{send} + {more} = {money}")

      print()
```

OUTPUT:

```
Solution found:
S=9,E=5,N=6,D=7,M=1,O=0,R=8,Y=2
9567 + 1085 = 10652
```

SLIP NO- 24

Q.1)Write a python program to sort the sentence in alphabetical order? [10 Marks]

ANS:

```
my_str="Hello this Is an Example With cased letters"

words=[word.lower() for word in my_str.split()]

words.sort()

print("The sorted words are:")

for word in words:

    print(word)
```

OUTPUT:

```
The sorted words are:
an
cased
example
hello
is
letters
this
with
```

Q.2) Write a Python program for the following Cryptorithmetic problems CROSS+ROADS = DANGER. [20Marks]

ANS:

```
import itertools

letters = ['C',  'R', 'O', 'S','A','D','N','G','E']

for perm in itertools.permutations(range(10), len(letters)):

    C,R,O,S,A,D,N,G,E  = perm

    if T == 0 or R == 0 or D==0:

        continue

    cross = 10000*C+1000*R+100*O+10 * S + S

    roads = 10000*R+1000*O+100*A+10 * D + S

    danger=100000*D+10000*A+1000*N+100*G+10 * E + R

    if cross+roads==danger:

        print(f"Solution found:")

        print(f"C={C},R={R}, O={O}, S={S},A={A},D={D}, N={N}, G={G},E={E}")

        print(f"{cross} + {roads} = {danger}")

        print()
```

OUTPUT:

```
Solution found:
C=9,R=6, O=2, S=3,A=5,D=1, N=8, G=7,E=4
96233 + 62513 = 158746
```

SLIP NO-25

Q.1). Build a bot which provides all the information related to you in college.

[10 Marks]

ANS:

```
from datetime import date, datetime
college_info = {
    "principal": "Dr.Mahamuni sir.",
    "courses": "We offer BSc Computer Science, BCom, BA, MSc, and other PG courses.",
    "departments": "We have departments like Computer Science, Commerce, Arts, Physics, Chemistry, and Mathematics.",
    "library": "Our library has more than 20,000 books, e-journals, and a digital study zone.",
    "events": "Annual Day, Tech Fest, Cultural Programs, and Sports Week are conducted every year.",
    "labs": "We have Computer Labs, Electronics Labs, and a Language Lab.",
    "canteen": "The college canteen provides hygienic and affordable food.",
    "date": f"Today is {date.today()}",
    "time": f"Current time is {datetime.now().strftime('%H:%M:%S')}",
    "bye": "Bye Bye! Have a great day!"
}


print("Hi, I am *College Info Bot v1.0*")
name = input("What is your name? => ")
print("Hi {}, Welcome to our college chatbot!".format(name))
print("You can ask me about: ")
for i in college_info.keys():
    print("-", i)
```

```
while True:
    query = input("\nEnter your question: ").lower()
    if query in college_info:
        print(college_info[query])
        if query == "bye":
            break
    else:
        print("Sorry, I don't have information about that. Please ask something else.")

    print("\nYou can ask me about:", ", ".join(college_info.keys()))
```

Q.2) Write a Python program to solve 8-puzzle problem. [20 Marks]

ANS:

```
import heapq
```

```
class PuzzleNode:
    def __init__(self, node_state, parent_node=None, move=None, cost=0):
        self.node_state = node_state
        self.parent_node = parent_node
        self.move = move
        self.cost = cost
        self.heuristic = self.calculate_heuristic()

    def __lt__(self, other):
        return (self.cost + self.heuristic) < (other.cost + other.heuristic)

    def calculate_heuristic(self):
        # Manhattan distance heuristic
        heuristic = 0
        for i in range(3):
            for j in range(3):
                if self.node_state[i][j] != 0:
                    goal_i, goal_j = divmod(self.node_state[i][j] - 1, 3)
                    heuristic += abs(i - goal_i) + abs(j - goal_j)
        return heuristic

def is_valid_move(i, j):
    return 0 <= i < 3 and 0 <= j < 3

def get_neighbors(node):
    neighbors = []
    i, j = next((i, j) for i in range(3) for j in range(3) if node.node_state[i][j] == 0)
    for di, dj in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        new_i, new_j = i + di, j + dj
        if is_valid_move(new_i, new_j):
            new_state = [row[:] for row in node.node_state]
            new_state[i][j], new_state[new_i][new_j] = new_state[new_i][new_j], new_state[i][j]
            neighbors.append(PuzzleNode(new_state, node, (i, j)))
    return neighbors

def reconstruct_path(node):
    path = []
    while node.parent_node is not None:
        path.append(node)
        node = node.parent_node
    return path[::-1]

def solve_8_puzzle(initial_state):
    start_node = PuzzleNode(initial_state)
    open_set = [start_node]
    closed_set = set()

    while open_set:
        current_node = heapq.heappop(open_set)
        if current_node.node_state == goal_state:
            return reconstruct_path(current_node)
        closed_set.add(tuple(map(tuple, current_node.node_state)))
        for neighbor in get_neighbors(current_node):
            if tuple(map(tuple, neighbor.node_state)) not in closed_set:
                heapq.heappush(open_set, neighbor)

    return None

if __name__ == "__main__":
    goal_state = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]  # Define the goal state
    initial_state = parse_input("__ed_input.txt")  # Parse initial state from input.txt
    path = solve_8_puzzle(initial_state)
    if path:
        print("Solution found! Moves to reach goal state:")
        for move in path:
            print(move.node_state)
    else:
        print("No solution found.")
```