Name :- Onasvee Banarse

Class:- TE Computer

ERP :-09

Subject :-LP2(AI) (BFS and DFS)

Code:-

```
import collections

# DFS algorithm
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start)

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited

# BFS algorithm
def bfs(graph, root):

    visited, queue = set(), collections.deque([root])
    visited.add(root)

    while queue:

        # Dequeue a vertex from queue
        vertex = queue.popleft()
        print(str(vertex) + " ", end="")

        # If not visited, mark it as visited, and
        # enqueue it
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)

vertex = []
Connections = []

no_vertex = int(input("Enter total number of vertex : "))
start_vertex = int(input("Enter starting vertex : "))

for i in range(no_vertex):
    vertex_n = int(input("Enter vertex " + str(i + 1) + " : "))
    # creating an empty list
    vertex.append(vertex_n)
    temp = []


    # number of elements as input
    n = int(input("Enter number of connections : "))

    # iterating till the range
    for i in range(0, n):
        ele = int(input("Enter connected to " + str(vertex_n) + " : "))
        temp.append(ele)  # adding the element

    print(temp)
    Connections.append(temp)


print(vertex)
print(Connections)
graph={ vertex[i]:Connections[i] for i in range(no_vertex)}
graph_dfs = {vertex[i]:set(Connections[i]) for i in range(no_vertex)}
print(graph)


flag = 1
while flag == 1:
    print("/*************MENU**************/")
    print("1. DFS")
    print("2. BFS ")
    print("3. Exit ")
    choice = int(input("Enter your choice : "))

    if choice == 1:
        print("Following is DFS :")
        print(dfs(graph_dfs, start_vertex))
    elif choice == 2:
        print("Following is BFS : " )
        print(bfs(graph, start_vertex))
    elif choice == 3:
        print("Exit")
        flag = 0
    else:
        print("Wrong Choice,Please Choose Another Option.")
```

Output:-

```
Enter total number of vertex : 4
Enter starting vertex : 2
Enter vertex 1 : 0
Enter number of connections : 2
Enter connected to 0 : 1
Enter connected to 0 : 2
[1, 2]
Enter vertex 2 : 1
Enter number of connections : 1
Enter connected to 1 : 2
[2]
Enter vertex 3 : 2
Enter number of connections : 2
Enter connected to 2 : 0
Enter connected to 2 : 3
[0, 3]
Enter vertex 4 : 3
Enter number of connections : 1
Enter connected to 3 : 3
[3]
[0, 1, 2, 3]
[[1, 2], [2], [0, 3], [3]]
{0: [1, 2], 1: [2], 2: [0, 3], 3: [3]}
/*************MENU**************/
1. DFS
2. BFS
3. Exit
Enter your choice : 1
Following is DFS :
2
```

```
0

1

3

/*************MENU**************/

1. DFS

2. BFS

3. Exit

Enter your choice : 2

Following is BFS :

2 0 3 1

/*************MENU**************/

1. DFS

2. BFS

3. Exit

Enter your choice : 5

Wrong Choice,Please Choose Another Option.

/*************MENU**************/

1. DFS

2. BFS

3. Exit

Enter your choice : 3

Exit


Process finished with exit code 0
```

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

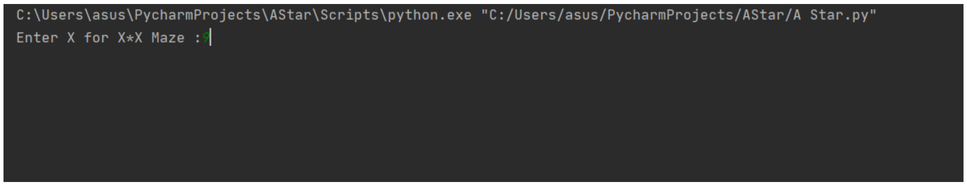
Subject :-LP2(AI) (A Star)

Code:-

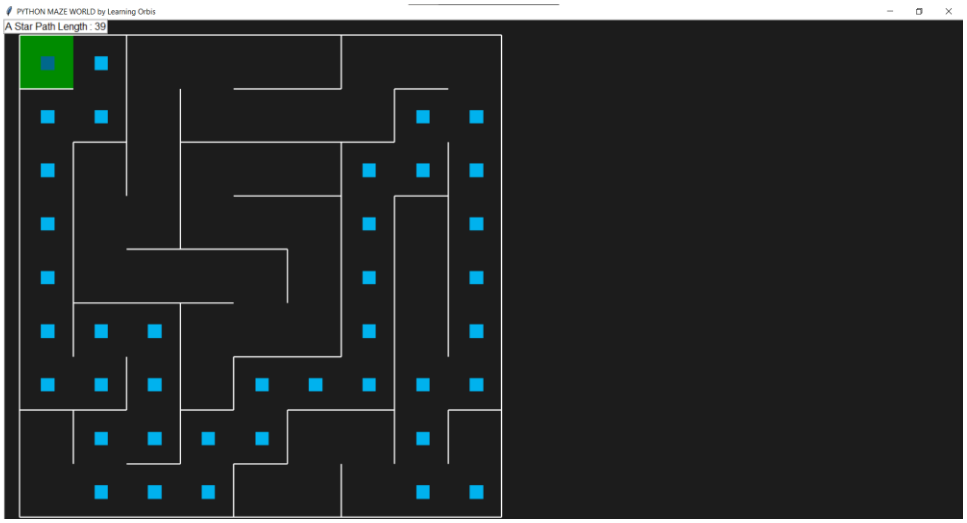
```
from pyamaze import maze,agent,textLabel
from queue import PriorityQueue
def h(cell1,cell2):
    x1,y1=cell1
    x2,y2=cell2

    return abs(x1-x2) + abs(y1-y2)
def aStar(m):
    start=(m.rows,m.cols)
    g_score={cell:float('inf') for cell in m.grid}
    g_score[start]=0
    f_score={cell:float('inf') for cell in m.grid}
    f_score[start]=h(start,(1,1))

    open=PriorityQueue()
    open.put((h(start,(1,1)),h(start,(1,1)),start))
    aPath={}
    while not open.empty():
        currCell=open.get()[2]
        if currCell==(1,1):
            break
        for d in 'ESNW':
            if m.maze_map[currCell][d]==True:
                if d=='E':
                    childCell=(currCell[0],currCell[1]+1)
                if d=='W':
                    childCell=(currCell[0],currCell[1]-1)
                if d=='N':
                    childCell=(currCell[0]-1,currCell[1])
                if d=='S':
                    childCell=(currCell[0]+1,currCell[1])

                temp_g_score=g_score[currCell]+1
                temp_f_score=temp_g_score+h(childCell,(1,1))

                if temp_f_score < f_score[childCell]:
                    g_score[childCell]= temp_g_score
                    f_score[childCell]= temp_f_score
                    open.put((temp_f_score,h(childCell,(1,1)),childCell))
                    aPath[childCell]=currCell
    fwdPath={}
    cell=(1,1)
    while cell!=start:
        fwdPath[aPath[cell]]=cell
        cell=aPath[cell]
    return fwdPath

if __name__=='__main__':
    x = int(input("Enter X for X*X Maze :"))
    m=maze(x,x)
    m.CreateMaze()
    path=aStar(m)

    a=agent(m,footprints=True)
    m.tracePath({a:path})
    l=textLabel(m,'A Star Path Length',len(path)+1)

    m.run()
```

Output:-

Enter X for X*X Maze :9



```
C:\Users\asus\PycharmProjects\AStar\Scripts\python.exe "C:/Users/asus/PycharmProjects/AStar/A Star.py"
Enter X for X*X Maze :9
```



A Star Path Length : 39

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

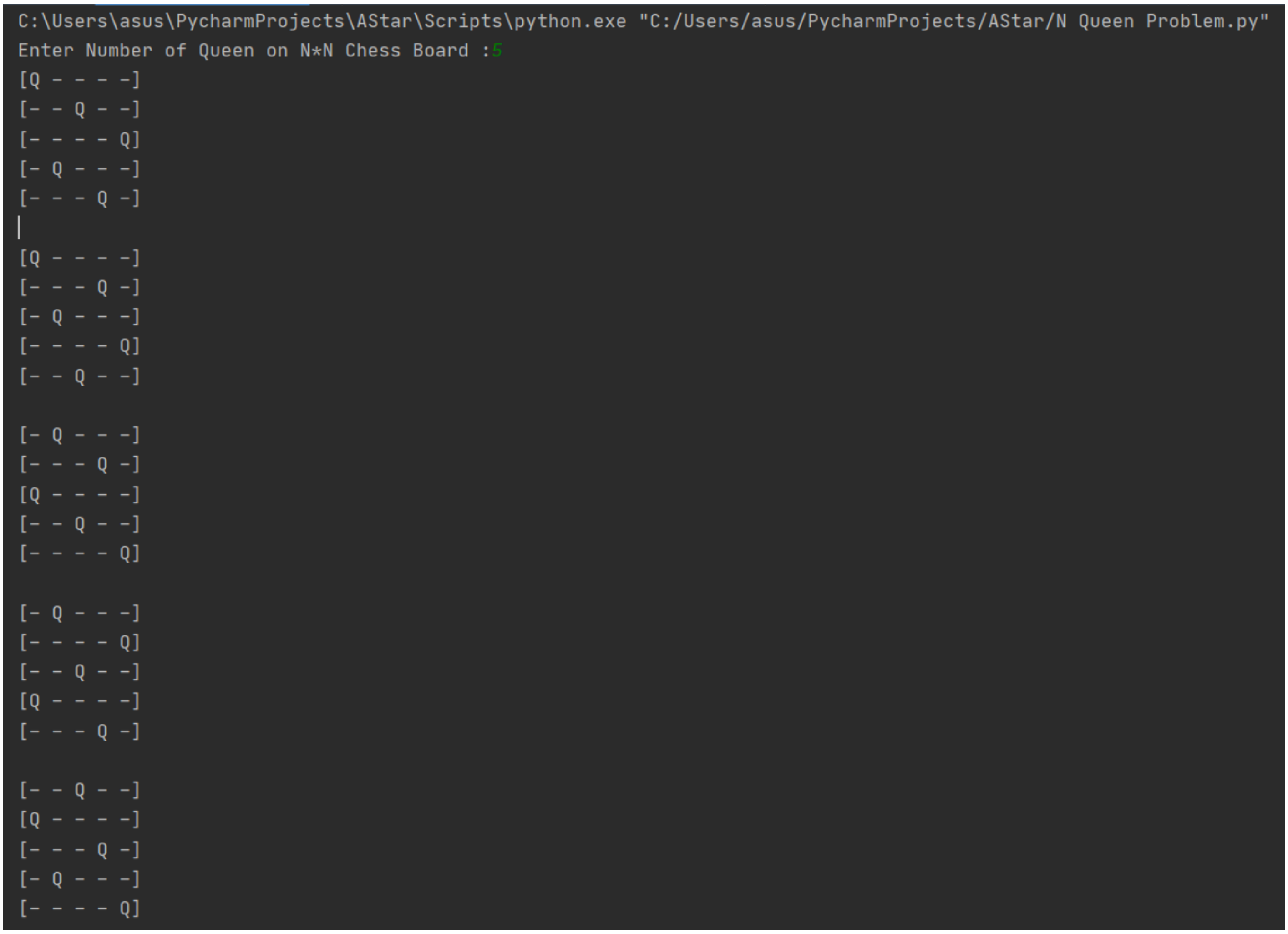
Subject :-LP2(AI) (N Queens)

Code:-

```
# Function to check if two queens threaten each other or not
def isSafe(mat, r, c):
    # return false if two queens share the same column
    for i in range(r):
        if mat[i][c] == 'Q':
            return False

    # return false if two queens share the same `` diagonal
    (i, j) = (r, c)
    while i >= 0 and j >= 0:
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j - 1

    # return false if two queens share the same `/` diagonal4
    (i, j) = (r, c)
    while i >= 0 and j < len(mat):
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j + 1

    return True


def printSolution(mat):
    for r in mat:
        print(str(r).replace(',', '').replace('\'', ''))
    print()


def nQueen(mat, r):
    # if `N` queens are placed successfully, print the solution
    if r == len(mat):
        printSolution(mat)
        return

    # place queen at every square in the current row `r`
    # and recur for each valid movement
    for i in range(len(mat)):

        # if no two queens threaten each other
        if isSafe(mat, r, i):
            # place queen on the current square
            mat[r][i] = 'Q'

            # recur for the next row
            nQueen(mat, r + 1)

            # backtrack and remove the queen from the current square
            mat[r][i] = '–'


if __name__ == '__main__':
    # `N × N` chessboard
    N = int(input("Enter Number of Queen on N*N Chess Board :"))

    # `mat[][]` keeps track of the position of queens in
    # the current configuration
    mat = [['–' for x in range(N)] for y in range(N)]

    nQueen(mat, 0)
```

Output:-

```
Enter Number of Queen on N*N Chess Board :5
[Q – – – –]
[– – Q – –]
[– – – – Q]
[– Q – – –]
[– – – Q –]

[Q – – – –]
[– – – Q –]
[– Q – – –]
[– – – – Q]
[– – Q – –]
[– – – – Q]

[– – Q – –]

[Q – – – –]

[– – – Q –]

[– Q – – –]


Process finished with exit code 0
```



```
C:\Users\asus\PycharmProjects\AStar\Scripts\python.exe "C:/Users/asus/PycharmProjects/AStar/N Queen Problem.py"
Enter Number of Queen on N*N Chess Board :5
[Q - - - -]
[- - Q - -]
[- - - - Q]
[- Q - - -]
[- - - Q -]

[Q - - - -]
[- - - Q -]
[- Q - - -]
[- - - - Q]
[- - Q - -]

[- Q - - -]
[- - - Q -]
[Q - - - -]
[- - Q - -]
[- - - - Q]

[- Q - - -]
[- - - - Q]
[- - Q - -]
[Q - - - -]
[- - - Q -]

[- - Q - -]
[Q - - - -]
[- - - Q -]
[- Q - - -]
[- - - - Q]
```

Name :- Onasvee Banarse

Class:- TE Computer

ERP :-09

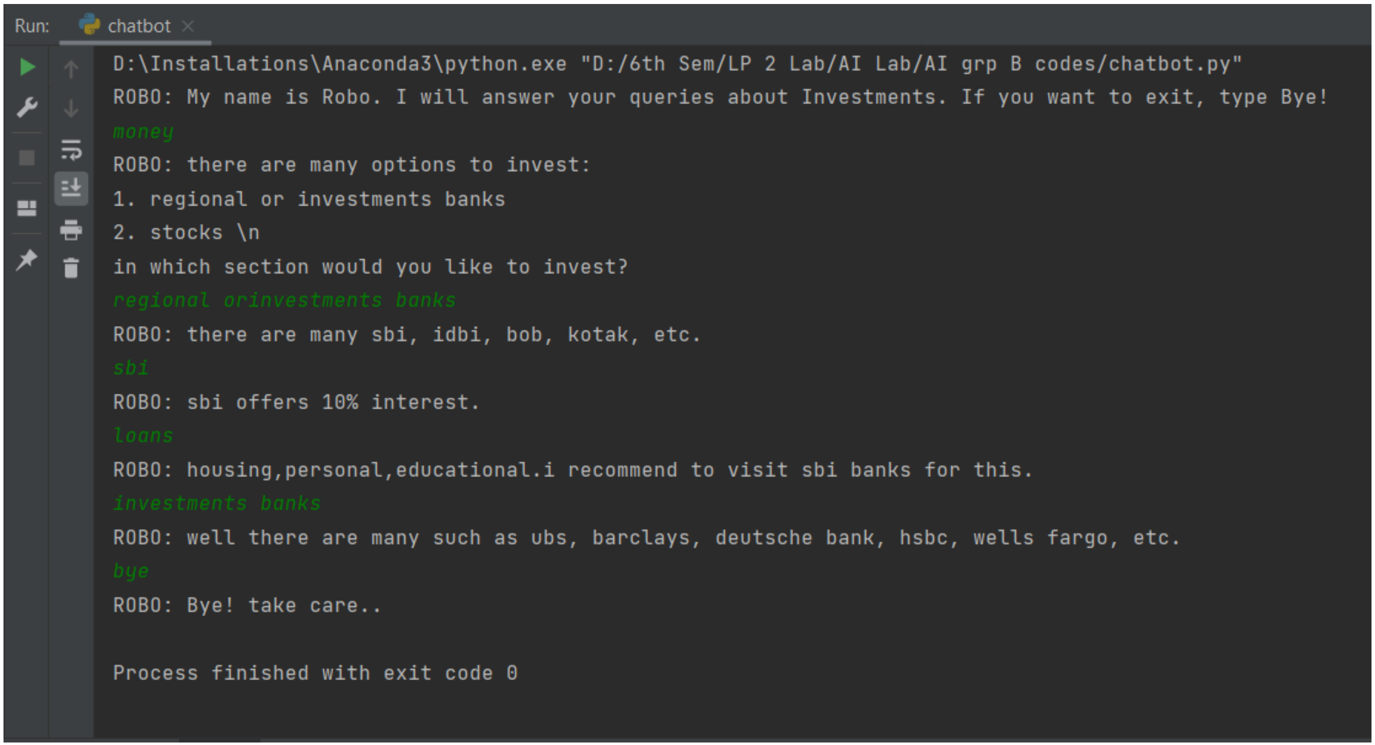
Subject :-LP2(AI) (Chatbot)

Code:-

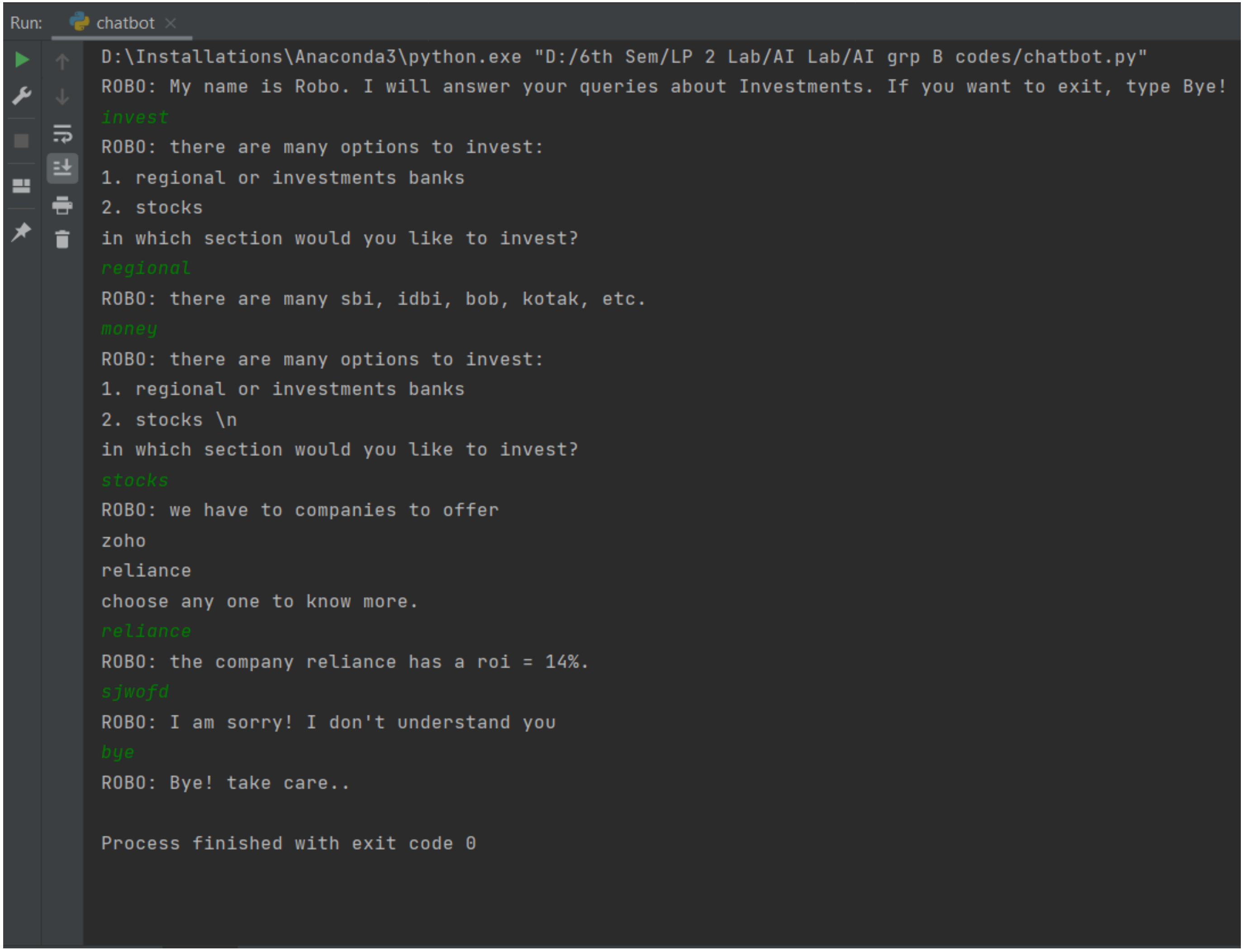
```
import io
import random
import string
import warnings
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import warnings
warnings.filterwarnings('ignore')
import nltk
from nltk.stem import WordNetLemmatizer
# nltk.download('popular', quiet=True)
# nltk.download('punkt')
# nltk.download('wordnet')

with open('chatbot.txt','r', encoding='utf8', errors ='ignore') as fin:
    raw = fin.read().lower()

#Tokenisation
sent_tokens = nltk.sent_tokenize(raw)
word_tokens = nltk.word_tokenize(raw)

# Preprocessing
lemmer = WordNetLemmatizer()
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))



# Keyword Matching
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey","Helo")
GREETING_RESPONSES = ["hi", "hey", "hi there", "hello", "I am glad! You are talking to me"]

def greeting(sentence):
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

```
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response

flag=True
print("ROBO: My name is Robo. I will answer your queries about Investments. If you want to exit, type Bye!")
while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("ROBO: You are welcome..")
        else:
            if(greeting(user_response)!=None):
                print("ROBO: "+greeting(user_response))
            else:
                print("ROBO: ",end="")
                res = response(user_response)
                nlines = res.count('\n')
                if nlines > 0:
                    res = res.split("\n",1)[1]
                print(res)
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("ROBO: Bye! take care..")
```

Output:-



```
D:\Installations\Anaconda3\python.exe "D:/6th Sem/LP 2 Lab/AI Lab/AI grp B codes/chatbot.py"
ROBO: My name is Robo. I will answer your queries about Investments. If you want to exit, type Bye!
money
ROBO: there are many options to invest:
1. regional or investments banks
2. stocks \n
in which section would you like to invest?
regional orinvestments banks
ROBO: there are many sbi, idbi, bob, kotak, etc.
sbi
ROBO: sbi offers 10% interest.
loans
ROBO: housing,personal,educational.i recommend to visit sbi banks for this.
investments banks
ROBO: well there are many such as ubs, barclays, deutsche bank, hsbc, wells fargo, etc.
bye
ROBO: Bye! take care..

Process finished with exit code 0
```



```
D:\Installations\Anaconda3\python.exe "D:/6th Sem/LP 2 Lab/AI Lab/AI grp B codes/chatbot.py"
ROBO: My name is Robo. I will answer your queries about Investments. If you want to exit, type Bye!
invest
ROBO: there are many options to invest:
1. regional or investments banks
2. stocks
in which section would you like to invest?
regional
ROBO: there are many sbi, idbi, bob, kotak, etc.
money
ROBO: there are many options to invest:
1. regional or investments banks
2. stocks \n
in which section would you like to invest?
stocks
ROBO: we have to companies to offer
zoho
reliance
choose any one to know more.
reliance
ROBO: the company reliance has a roi = 14%.
sjwofd
ROBO: I am sorry! I don't understand you
bye
ROBO: Bye! take care..

Process finished with exit code 0
```

Name :- Onasvee Banarse

Class:- TE Computer

ERP :-09

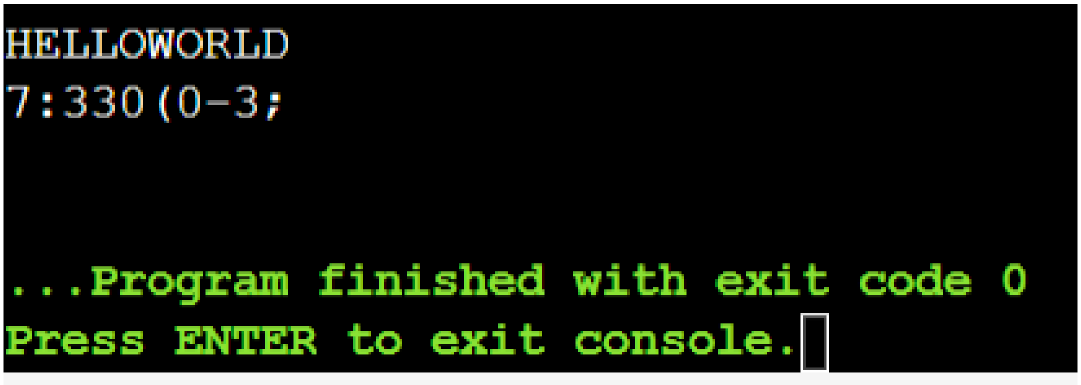
Subject :-LP2(IS) (Logical Operations)

Code:-

```
#include <iostream.h>
//using namespace std;
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
   //clrscr();
   char str[]="HELLOWORLD";
   char str1[11];
   char str2[11];
   int i,len;
   len = strlen(str);

   for(i=0;i<len;i++)
   {
      str1[i]=str[i] & 127;
      cout<<str1[i];
   }
   cout<<"\n";
   for(i=0;i<len;i++)
   {
      str2[i] = str[i] ^ 127;
      cout<<str2[i];
   }
```

```
    cout<<"\n";
    getch();
}
```

Output:-



```
HELLOWORLD
7:330(0-3;


...Program finished with exit code 0
Press ENTER to exit console.
```

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

Subject :-LP2(IS) (Transposition)

Code:-

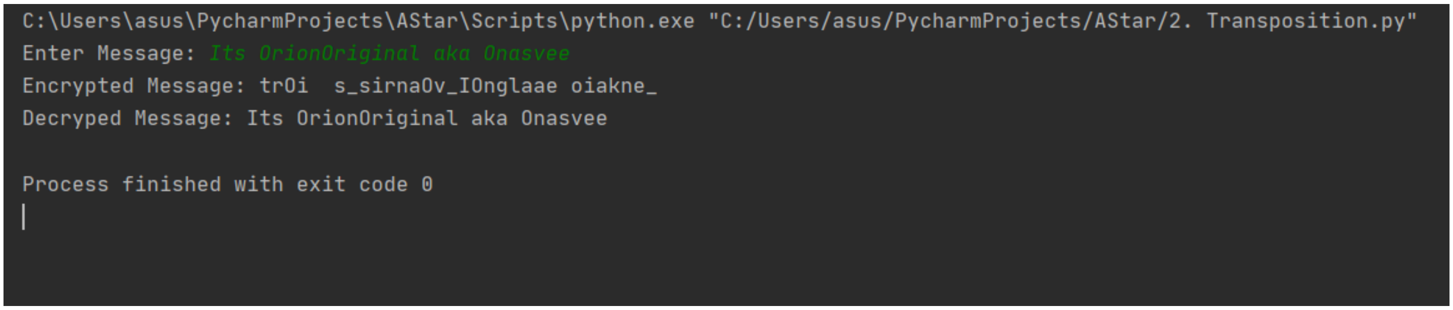
```
import math

key = "HACK"

# Encryption
def encryptMessage(msg):
    cipher = ""

   # track key indices
    k_indx = 0

   msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))

   # calculate column of the matrix
    col = len(key)

   # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))

   # add the padding character '_' in empty
    # the empty cell of the matix
   fill_null = int((row * col) - msg_len)
    msg_lst.extend('_' * fill_null)

   # create Matrix and insert message and
   # padding characters row-wise
   matrix = [msg_lst[i: i + col]
             for i in range(0, len(msg_lst), col)]

   # read matrix column-wise using key
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ".join([row[curr_idx]
                         for row in matrix])
        k_indx += 1

   return cipher
```

```
# Decryption
def decryptMessage(cipher):
    msg = ""

   # track key indices
    k_indx = 0

   # track msg indices
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)

   # calculate column of the matrix
    col = len(key)

   # calculate maximum row of the matrix
    row = int(math.ceil(msg_len / col))

   # convert key into list and sort
   # alphabetically so we can access
   # each character by its alphabetical position.
    key_lst = sorted(list(key))

   # create an empty matrix to
   # store deciphered message
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]

   # Arrange the matrix column wise according
   # to permutation order by adding into new matrix
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])

       for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1

   # convert decrypted msg matrix into a string
    try:
        msg = ".join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")

   null_count = msg.count('_')
```

```
    if null_count > 0:
        return msg[: -null_count]

    return msg

# Driver Code

msg = (input("Enter Message: "))


cipher = encryptMessage(msg)
print("Encrypted Message: {}".
                format(cipher))

print("Decryped Message: {}".
        format(decryptMessage(cipher)))
```

Output:-

```
Enter Message: Its OrionOriginal aka Onasvee

Encrypted Message: trOi  s_sirnaOv_IOnglaae oiakne_

Decryped Message: Its OrionOriginal aka Onasvee


Process finished with exit code 0
```



```
C:\Users\asus\PycharmProjects\AStar\Scripts\python.exe "C:/Users/asus/PycharmProjects/AStar/2. Transposition.py"
Enter Message: Its OrionOriginal aka Onasvee
Encrypted Message: trOi  s_sirnaOv_IOnglaae oiakne_
Decryped Message: Its OrionOriginal aka Onasvee

Process finished with exit code 0
```

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

Subject :-LP2(IS) (AES)

Code:-

```
import hashlib
from base64 import b64decode, b64encode

from Crypto import Random
from Crypto.Cipher import AES


class AESCipher(object):
    def __init__(self, key):
        self.block_size = AES.block_size
        self.key = hashlib.sha256(key.encode()).digest()

    def encrypt(self, plain_text):
        plain_text = self.__pad(plain_text)
        iv = Random.new().read(self.block_size)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        encrypted_text = cipher.encrypt(plain_text.encode())
        return b64encode(iv + encrypted_text).decode("utf-8")

    def decrypt(self, encrypted_text):
        encrypted_text = b64decode(encrypted_text)
        iv = encrypted_text[:self.block_size]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        plain_text = cipher.decrypt(encrypted_text[self.block_size:]).decode("utf-8")
        return self.__unpad(plain_text)

    def __pad(self, plain_text):
        number_of_bytes_to_pad = self.block_size - len(plain_text) % self.block_size
        ascii_string = chr(number_of_bytes_to_pad)
        padding_str = number_of_bytes_to_pad * ascii_string
        padded_plain_text = plain_text + padding_str
        return padded_plain_text

    @staticmethod
    def __unpad(plain_text):
        last_character = plain_text[len(plain_text) - 1:]
        return plain_text[:-ord(last_character)]


key = input("Enter Key: ")
```

```
aes = AESCipher(key)

flag = 1
while flag == 1:
    print("/*************MENU**************/")
    print("1. Encryption")
    print("2. Decryption")
    print("3. Exit ")
    choice = int(input("Enter your choice : "))

    if choice == 1:
        message = input("Enter message to encrypt: ")
        encryptedMessage = aes.encrypt(message)
        print("Encrypted Message:", encryptedMessage)

    elif choice == 2:
        message = input("Enter message to decrypt: ")
        decryptedMessage = aes.decrypt(message)
        print("Decrypted Message:", decryptedMessage)
    elif choice == 3:
        print("Exit")
        flag = 0
    else:
        print("Wrong Choice,Please Choose Another Option.")
```

Output:-

```
Enter Key: AISSMSIOIT

/*************MENU**************/

1. Encryption

2. Decryption

3. Exit

Enter your choice : 1

Enter message to encrypt: Its OrionOrignal aka Onasvee

Encrypted Message:
icrRcUjOKrKfNzmQF1YTnCMuXsILZjhbdtSCA84WuzKT21T1lYiYCyx4IayIfdR5

/*************MENU**************/

1. Encryption
```

```
2. Decryption

3. Exit

Enter your choice : 2

Enter message to decrypt:
icrRcUjOKrKfNzmQF1YTnCMuXsILZjhbdtSCA84WuzKT21T1lYiYCyx4IayIfdR5

Decrypted Message: Its OrionOrignal aka Onasvee

/*************MENU**************/

1. Encryption

2. Decryption

3. Exit

Enter your choice : 4

Wrong Choice,Please Choose Another Option.

/*************MENU**************/

1. Encryption

2. Decryption

3. Exit

Enter your choice : 3

Exit


Process finished with exit code 0
```

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

Subject :-LP2(IS) (DES)

Code:-

```
# Hexadecimal to binary conversion
def hex2bin(s):
    mp = {'0' : "0000",
        '1' : "0001",
        '2' : "0010",
        '3' : "0011",
        '4' : "0100",
        '5' : "0101",
        '6' : "0110",
        '7' : "0111",
        '8' : "1000",
        '9' : "1001",
        'A' : "1010",
        'B' : "1011",
        'C' : "1100",
        'D' : "1101",
        'E' : "1110",
        'F' : "1111" }
    bin = ""
    for i in range(len(s)):
        bin = bin + mp[s[i]]
    return bin

# Binary to hexadecimal conversion
def bin2hex(s):
    mp = {"0000" : '0',
        "0001" : '1',
        "0010" : '2',
        "0011" : '3',
        "0100" : '4',
        "0101" : '5',
        "0110" : '6',
        "0111" : '7',
        "1000" : '8',
        "1001" : '9',
        "1010" : 'A',
        "1011" : 'B',
        "1100" : 'C',
        "1101" : 'D',
        "1110" : 'E',
      "1111" : 'F' }
   hex = ""
   for i in range(0,len(s),4):
      ch = ""
      ch = ch + s[i]
      ch = ch + s[i + 1]
      ch = ch + s[i + 2]
      ch = ch + s[i + 3]
      hex = hex + mp[ch]

  return hex

# Binary to decimal conversion
def bin2dec(binary):

  binary1 = binary
   decimal, i, n = 0, 0, 0
   while(binary != 0):
      dec = binary % 10
      decimal = decimal + dec * pow(2, i)
      binary = binary//10
      i += 1
   return decimal

# Decimal to binary conversion
def dec2bin(num):
   res = bin(num).replace("0b", "")
   if(len(res)%4 != 0):
      div = len(res) / 4
      div = int(div)
      counter =(4 * (div + 1)) - len(res)
      for i in range(0, counter):
         res = '0' + res
   return res

# Permute function to rearrange the bits
def permute(k, arr, n):
   permutation = ""
   for i in range(0, n):
      permutation = permutation + k[arr[i] - 1]
   return permutation

# shifting the bits towards left by nth shifts
def shift_left(k, nth_shifts):
   s = ""
   for i in range(nth_shifts):
      for j in range(1,len(k)):
         s = s + k[j]
```

```
        s = s + k[0]
        k = s
        s = ""
    return k

# calculating xow of two strings of binary number a and b
def xor(a, b):
    ans = ""
    for i in range(len(a)):
        if a[i] == b[i]:
            ans = ans + "0"
        else:
            ans = ans + "1"
    return ans

# Table of Position of 64 bits at initial level: Initial Permutation Table
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]

# Expansion D-box Table
exp_d = [32, 1 , 2 , 3 , 4 , 5 , 4 , 5,
        6 , 7 , 8 , 9 , 8 , 9 , 10, 11,
        12, 13, 12, 13, 14, 15, 16, 17,
        16, 17, 18, 19, 20, 21, 20, 21,
        22, 23, 24, 25, 24, 25, 26, 27,
        28, 29, 28, 29, 30, 31, 32, 1 ]

# Straight Permutation Table
per = [ 16, 7, 20, 21,
        29, 12, 28, 17,
        1, 15, 23, 26,
        5, 18, 31, 10,
        2, 8, 24, 14,
        32, 27, 3, 9,
        19, 13, 30, 6,
        22, 11, 4, 25 ]

# S-box Table
sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
        [ 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
        [ 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
        [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 ]],
       [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
            [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
            [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
        [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 ]],

        [ [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
        [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
        [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
            [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 ]],

       [ [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
        [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
        [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
            [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14] ],

       [ [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
        [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
            [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
        [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 ]],

       [ [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
        [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
            [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
            [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13] ],

       [ [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
        [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
            [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
            [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12] ],

       [ [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
            [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
            [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
            [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11] ] ]

# Final Permutation Table
final_perm = [ 40, 8, 48, 16, 56, 24, 64, 32,
            39, 7, 47, 15, 55, 23, 63, 31,
            38, 6, 46, 14, 54, 22, 62, 30,
            37, 5, 45, 13, 53, 21, 61, 29,
            36, 4, 44, 12, 52, 20, 60, 28,
            35, 3, 43, 11, 51, 19, 59, 27,
            34, 2, 42, 10, 50, 18, 58, 26,
            33, 1, 41, 9, 49, 17, 57, 25 ]

def encrypt(pt, rkb, rk):
    pt = hex2bin(pt)
    # Initial Permutation
     pt = permute(pt, initial_perm, 64)
     print("After initial permutation", bin2hex(pt))

    # Splitting
     left = pt[0:32]
     right = pt[32:64]
     for i in range(0, 16):
        # Expansion D-box: Expanding the 32 bits data into 48 bits
        right_expanded = permute(right, exp_d, 48)

        # XOR RoundKey[i] and right_expanded
        xor_x = xor(right_expanded, rkb[i])

        # S-boxex: substituting the value from s-box table by calculating row and column
        sbox_str = ""
        for j in range(0, 8):
           row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
           col = bin2dec(int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] + xor_x[j * 6 + 3] + xor_x[j * 6 + 4]))
           val = sbox[j][row][col]
           sbox_str = sbox_str + dec2bin(val)

        # Straight D-box: After substituting rearranging the bits
        sbox_str = permute(sbox_str, per, 32)

        # XOR left and sbox_str
        result = xor(left, sbox_str)
        left = result

        # Swapper
        if(i != 15):
           left, right = right, left
        print("Round ", i + 1, " ", bin2hex(left), " ", bin2hex(right), " ", rk[i])

    # Combination
     combine = left + right

    # Final permutation: final rearranging of bits to get cipher text
     cipher_text = permute(combine, final_perm, 64)
     return cipher_text

pt = "123456ABCD132536"
key = "AABB09182736CCDD"

# Key generation
# --hex to binary
key = hex2bin(key)
```

```
# --parity bit drop table
keyp = [57, 49, 41, 33, 25, 17, 9,
      1, 58, 50, 42, 34, 26, 18,
      10, 2, 59, 51, 43, 35, 27,
      19, 11, 3, 60, 52, 44, 36,
      63, 55, 47, 39, 31, 23, 15,
      7, 62, 54, 46, 38, 30, 22,
      14, 6, 61, 53, 45, 37, 29,
      21, 13, 5, 28, 20, 12, 4 ]

# getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56)

# Number of bit shifts
shift_table = [1, 1, 2, 2,
            2, 2, 2, 2,
            1, 2, 2, 2,
            2, 2, 2, 1 ]

# Key- Compression Table : Compression of key from 56 bits to 48 bits
key_comp = [14, 17, 11, 24, 1, 5,
         3, 28, 15, 6, 21, 10,
         23, 19, 12, 4, 26, 8,
         16, 7, 27, 20, 13, 2,
         41, 52, 31, 37, 47, 55,
         30, 40, 51, 45, 33, 48,
         44, 49, 39, 56, 34, 53,
         46, 42, 50, 36, 29, 32 ]

# Splitting
left = key[0:28] # rkb for RoundKeys in binary
right = key[28:56] # rk for RoundKeys in hexadecimal

rkb = []
rk = []
for i in range(0, 16):
   # Shifting the bits by nth shifts by checking from shift table
   left = shift_left(left, shift_table[i])
   right = shift_left(right, shift_table[i])

  # Combination of left and right string
   combine_str = left + right

  # Compression of key from 56 to 48 bits
   round_key = permute(combine_str, key_comp, 48)

   rkb.append(round_key)
```

```
    rk.append(bin2hex(round_key))

print("Encryption")
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ",cipher_text)

print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ",text)
```

Output:-

```
Encryption
After initial permutation 14A7D67818CA18AD
Round  1   18CA18AD   5A78E394   194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
Round  9   308BEE97   10AF9D37   84BB4473DCCC
Round  10   10AF9D37   6CA6CB20   02765708B5BF
Round  11   6CA6CB20   FF3C485F   6D5560AF7CA5
Round  12   FF3C485F   22A5963B   C2C1E96A4BF3
Round  13   22A5963B   387CCDAA   99C31397C91F
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text :  C0B7A8D05F3A829C
```

```
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536

Process finished with exit code 0
```

Name :- Onasvee Banarse

CLass:- TE Computer

ERP :-09

Subject :-LP2(IS) (RSA)

Code:-

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii

msg = (input("Enter Message to Encrypt and Decrypt : "))
msg = bytes(msg, 'utf-8')

keyPair = RSA.generate(3072)

pubKey = keyPair.publickey()
print(f"Public key:  (n={hex(pubKey.n)}, e={hex(pubKey.e)})")
pubKeyPEM = pubKey.exportKey()
print(pubKeyPEM.decode('ascii'))

print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")
privKeyPEM = keyPair.exportKey()
print(privKeyPEM.decode('ascii'))

# msg = input()
encryptor = PKCS1_OAEP.new(pubKey)
encrypted = encryptor.encrypt(msg)
print("Encrypted:", binascii.hexlify(encrypted))

decryptor = PKCS1_OAEP.new(keyPair)
decrypted = decryptor.decrypt(encrypted)
print('Decrypted:', decrypted)
```

Output:-

```
Enter Message to Encrypt and Decrypt : Its OrionOriginal aka Onasvee

Public key:
(n=0xd28fb8466404a25918fa62ceb454a7a2ecc4fa1fab0ef0ab3ce5afb29499cac1c860765a5680d6dfb598588e
7f75e1996ca74636ff0b3d5c18679f9609c94645e3157b330b1e4aba50f7a990e58cddd5a0d1521c9b772e6c3c0
d72f721a74495da5874e12d7fe5bd1a9f419b0cfc52a77597a51fbf687186029b323d97540281c2f954573480a81
071bff175298ae97371cb700c3ff4dc5afab62799490fd2259e648bc2af0d6163f3c533558cfef08e1d5bd3d7d238
c9e279ffd50c555ca9e11865fa7bfc8088fed2fe6b0ecdab26621f1e08734d7f58634c628d3989663afbb6c2f89c4d
cd4042652d5ca23aa3b90ab7d0f3582c011130b890f7d106466f8e1ac4ff5ec55c3f1ad6c5727f34afb39f2559082
b0156a569d1a449a8ceda4aa656ba61ab1963df7c3cd64cdcb013e28bfff416419fb94406a15dc08c6ef97deb9e8
```

30221fc321fc3d65ac669a101ce754f81b5e6e6e7d7768646e90620801906f646ce7cdcec421e4c0c166a6b57023
8118ba9fb53acbe8e5ae85ca270ca48fb, e=0x10001)

-----BEGIN PUBLIC KEY-----

MIIBojANBgkqhkiG9w0BAQEFAAOCAY8AMIIBigKCAYEA0o+4RmQEolkY+mLOtFSn
ouzE+h+rDvCrPOWvspSZysHIYHZaVoDW37WYWI5/deGZbKdGNv8LPVwYZ5+WCclG
ReMVezMLHkq6UPepkOWM3dWg0VIcm3cubDwNcvchp0SV2lh04S1/5b0an0GbDPxS
p3WXpR+/aHGGApsyPZdUAoHC+VRXNICoEHG/8XUpiulzcctwDD/03Fr6tieZSQ/S
JZ5ki8KvDWFj88UzVYz+8I4dW9PX0jjJ4nn/1QxVXKnhGGX6e/yAiP7S/msOzasm
Yh8eCHNNf1hjTGKNOYlmOvu2wvicTc1AQmUtXKI6o7kKt9DzWCwBETC4kPfRBkZv
jhrE/17FXD8a1sVyfzSvs58lWQgrAValadGkSajO2kqmVrphqxlj33w81kzcsBPi
i//0FkGfuUQGoV3AjG75feuegwIh/DIfw9ZaxmmhAc51T4G15ubn13aGRukGIIAZ
BvZGznzc7EIeTAwWamtXAjgRi6n7U6y+jlroXKJwykj7AgMBAAE=

-----END PUBLIC KEY-----

Private key:
(n=0xd28fb8466404a25918fa62ceb454a7a2ecc4fa1fab0ef0ab3ce5afb29499cac1c860765a5680d6dfb598588e
7f75e1996ca74636ff0b3d5c18679f9609c94645e3157b330b1e4aba50f7a990e58cddd5a0d1521c9b772e6c3c0
d72f721a74495da5874e12d7fe5bd1a9f419b0cfc52a77597a51fbf687186029b323d97540281c2f954573480a81
071bff175298ae97371cb700c3ff4dc5afab62799490fd2259e648bc2af0d6163f3c533558cfef08e1d5bd3d7d238
c9e279ffd50c555ca9e11865fa7bfc8088fed2fe6b0ecdab26621f1e08734d7f58634c628d3989663afbb6c2f89c4d
cd4042652d5ca23aa3b90ab7d0f3582c011130b890f7d106466f8e1ac4ff5ec55c3f1ad6c5727f34afb39f2559082
b0156a569d1a449a8ceda4aa656ba61ab1963df7c3cd64cdcb013e28bfff416419fb94406a15dc08c6ef97deb9e8
30221fc321fc3d65ac669a101ce754f81b5e6e6e7d7768646e90620801906f646ce7cdcec421e4c0c166a6b57023
8118ba9fb53acbe8e5ae85ca270ca48fb,
d=0x2d24d92a665944017c447a98bcbb05b1fdb781b4f674de8ea820ca99ac18890b210de5721ae7c6a9f20236c
25e7b84a1e354bdce1ec267266ea910e317380b1402cae13e215d1e4272079758548eee24d634eab8ed701108
ed9b2891e9aa361f36d00e4714fd3de15c6ad6a30a96b295eab55796c5effb9ef2c21974711476f1213f59a0d4c5
dcc2a1d0b85119560a1551497fbd709cebfda991124e6006bf5487702132dd5b2e0d42ff7db112e8b9e48e50d8c
b85ebdd04ec8938414baff14fc8b8c364f96e242c821fb2ceb8815dc445bb9100797b693ec330d7815044276cf97
7be385f8f23c837848024224a7070d2cf4773588cbe57ecdcffeec2a5fa0c0e0ff4e5829221777dce02ae1828fffc3
4369e5157fe3fcf1066a3132d2d7182aac909eb3dbf2f5cc297762896ec4cfb149ad2b879667960117c80f65ff1c8
7d1ba0831761676d4201bf042e94c49ccd8b797bbc79e46d3a55f9b25d2632d3d4a352beee58ed56d1e9debcff
b6febdcf44f977ac43f421415a83a9bbe0352981)

-----BEGIN RSA PRIVATE KEY-----

MIIG5AIBAAKCAYEA0o+4RmQEolkY+mLOtFSnouzE+h+rDvCrPOWvspSZysHIYHZa
VoDW37WYWI5/deGZbKdGNv8LPVwYZ5+WCclGReMVezMLHkq6UPepkOWM3dWg0VIc
m3cubDwNcvchp0SV2lh04S1/5b0an0GbDPxSp3WXpR+/aHGGApsyPZdUAoHC+VRX
NICoEHG/8XUpiulzcctwDD/03Fr6tieZSQ/SJZ5ki8KvDWFj88UzVYz+8I4dW9PX

0jjJ4nn/1QxVXKnhGGX6e/yAiP7S/msOzasmYh8eCHNNf1hjTGKNOYlmOvu2wvic
Tc1AQmUtXKI6o7kKt9DzWCwBETC4kPfRBkZvjhrE/17FXD8a1sVyfzSvs58lWQgr
AValadGkSajO2kqmVrphqxlj33w81kzcsBPii//0FkGfuUQGoV3AjG75feuegwIh
/DIfw9ZaxmmhAc51T4G15ubn13aGRukGIIAZBvZGznzc7EIeTAwWamtXAjgRi6n7
U6y+jlroXKJwykj7AgMBAAECggGALSTZKmZZRAF8RHqYvLsFsf23gbT2dN6OqCDK
mawYiQshDeVyGufGqfICNsJee4Sh41S9zh7CZyZuqRDjFzgLFALK4T4hXR5CcgeX
WFSO7iTWNOq47XARCO2bKJHpqjYfNtAORxT9PeFcatajCpayleq1V5bF7/ue8sIZ
dHEUdvEhP1mg1MXcwqHQuFEZVgoVUUl/vXCc6/2pkRJOYAa/VIdwITLdWy4NQv99
sRLoueSOUNjLhevdBOyJOEFLr/FPyLjDZPluJCyCH7LOuIFdxEW7kQB5e2k+wzDX
gVBEJ2z5d744X48jyDeEgCQiSnBw0s9Hc1iMvlfs3P/uwqX6DA4P9OWCkiF3fc4C
rhgo//w0Np5RV/4/zxBmoxMtLXGCqskJ6z2/L1zCl3YoluxM+xSa0rh5ZnlgEXyA
9l/xyH0boIMXYWdtQgG/BC6UxJzNi3l7vHnkbTpV+bJdJjLT1KNSvu5Y7VbR6d68
/7b+vc9E+XesQ/QhQVqDqbvgNSmBAoHBANuhVjk2UR2Q+HZtolCu0Bm4BGjgmHSm
Juho6+kGSiRS5Hg7hRccm+fjQd0kGk0yJTcHU8g5UF+hlMT7jHhQlcNWttQVVvdc
SWgQaoEUcZnpAyFMqTL4eNezCHOyvnRh4h2RFoxrDPE5DlPVQab+GSVMorHp28g0
v5WwrSHdjcdoXXObgCWlq5WfbsshQuJF1FhsTtdsb5Y/JAvoDpi76tAIbvhrr0u0
SouZv7+ucH/bFJFwFNRVHw0ZCQPNVlEvOwKBwQD1be8FA4cUgtGU7UrCLmFwjI/f
KGgfLgPUafU63yneMgPajLKu7acBz24eY+ab0etBfeF371YSkKFLA6/7wUHOQw/r
PMKg77AWCu01VV5T6yX5YE081tOCSHtziy9OFYLdd/z408pweiHOhPD02Gz8V+50
etE0zaQ9dYj86FL7PPGO0Fxp0bGJDl14nvJoKfCiIFSRjZVnTJGw98TaIuurRmZ9
MQKP9fbdtpcxDQ6mZqFSGsALDykyqcuXu+g5MUECgcEAjpvR2tBUFzicvHkvnegE
o86Cvn6nP4brWJlYJTS6S5+vTgqHvpwK96TujWL12Q4ob/TICAh/EblfWhBkA3N/
6xiRGmDI2VEJMRMHtMzLfr54E9UtQDVqcdSENmvnkrZEFiKxW3ffLXp4vSKJwJ7Z
QQjj01YgKX1msRHJOWYcu1Ae7gQYT1mlcj/Vtvvf7ACfgtLA1sxIIGzbQQfrAm1y
aKYxOAjkB+oHRWINya7AyaQ9VLpMLBshUGXjHp7j308lAoHBAK/GOU509VSqUKIB
xO4Hu7+Y3B2uWcwi75k8/eZZGCpL1di7tel0yYyRXEOltu7YTE5OcqGsJxAKx4nr
LSn4gkHQY+FNVfNfVtSipLry1ijyG/NbllXBYiBH+yqIf6vD2kL1gZdQUAd4YSgA
rHYfXwbnjx+bKqRPt5ZQzHidh3jqb/Khpd4f0a/gOu99nw0dJHto/kh0h5FBFIMT
IMg+BF1ZgWOeK0Chn1mxQN1fhaOFk3ozMGF7TT08wFR+vtXfQQKBwA8bkaOFL7e4
9vw75920f8CXFLoP5uOEOvTvucFcpLyJ45WWE5o98N9mNH093gW9oNLCPFcr7eiD

```
dl6WtjbS82UXMlDGDJv+7al71AY/wTQbeMvqKKtzzBXeoNADcXsVRsqbBB7KPJb4

uJq6Rr9rPzcv8TeIPXG7t0FS4V7jwwgCdFsOT6M1Zh9dLwdsSydWcyl3N4x1gQzE

PrHA1jVVkPdQZ5492piZo6V1XMS36F9FiC9IyxJ1840LwTV/rvtsnw==

-----END RSA PRIVATE KEY-----

Encrypted:
b'cb47be1bba8b1153b1a2ba8edd59e87bcaf3c764749ce049ae3b1c3e3c69e8a441750f12109e50e4c2295d518
9a337f600e4563918c903c014fb5ea63d4ad0e99bdf83ccb4ce7e5a9437044f58b88568e59071895c128288cbfb
9136b287da8e1abd1bee1f1104877f2bcf8db12fa80018f1f7a1afcd29f6bb405e152b8eae65746dc26f87e87b2e
a7ac8e5ed06df14053b597ff53b33bf00be482b34f24ebeb5a3b4b6290ba700c86e9fd6517d3aab06cd8fa403c0e
1dcd1e8790b886a8a50453656cc8feb46bd7fdd9ba31781b98362af57e9f13a159c2fdcac54ce9e5d05cf578c935
5603fe4c1e6e6aefd8939f25eeb1a4472d8ccfd513c1dab0c1772bdc6de6af67c623d6d89f8534d11088e4267dc5
8101db696d4ef8b48228e363e1831ab8850561b3e54d605b4efb0b903767002e95276e22279dd02e17e4b489ff
3e9938fdc16694097fcbd4692fa5660d4bbec55d6aa3342f31f52b3744a0a75b3dabe271eaf04a182feb49bb2480
da6226de00b8bfcef02654d88589b94442b450'

Decrypted: b'Its OrionOriginal aka Onasvee'


Process finished with exit code 0
```