

NAME : AVISHKAAR PAWAR

ROLL NO : AD-1224

Course : B.Sc(H) CS

Subject : Artificial Intelligence (GE-II)

Batch : 2021-2024

Submitted to : Ms. Monika Bhattacharya

LAB 1

#Question 1

#To accept the ‘name’ of the user and print Welcome message

name=input("Enter Your name : ")

print("Hello ",name)

#OUTPUT#

Enter Your name : X

Hello X

#Question 2

#To accept real (decimal) numbers from the user and print their addition

ls=eval(input("Enter numbers to be added in list form : "))

def add(ls):

  sum=0

  for a in ls:

    sum+=a

  return sum

print("Sum is = ",add(ls))

#OUTPUT#

Enter numbers to be added in list form : [3,7,100]

Sum is = 110

# Question 3

#To accept the value of current temperature from the user.

#( If T > 27 ⁰C, print weather is HOT, If T = 27 ⁰C, print weather is COMFORTABLE and If T < 27 ⁰C, print weather is COOL)

temp=int(input("Enter Temperature : "))

if (temp>27):

    print("Weather is HOT")

elif (temp==27):

    print("Weather is Comfortable")

else :

    print("Weather is COOL")

#OUTPUT#

Enter Temperature : 28

Weather is HOT

#Question 4

#Calculator

lst=[]

def menu():

    print("\n-----------------------------------")

    print("| 1: ADD         2:Subtract       |\n|3: MULTIPLY , 4:DIVIDE , 5:Break |")

    print("-----------------------------------\n")

    num1=float(input("Enter First Number "))

    num2=float(input("Enter Second Number "))

    op=int(input("Enter your choice : "))

    if (op==1):

        print("num1 + num2 = ",num1+num2)

    elif (op==2):

      print("num1 - num 2 = ",num1-num2)

    elif (op==3):

      print("num1 \* num 2 = ",num1\*num2)

    elif (op==4):

      print("num1 / num 2 = ",num1/num2)

    elif (op==5):

      return

    menu()

menu()

print("Program Finished Successfully ")

#OUTPUT#

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 12

Enter Second Number 4

Enter your choice : 1

num1 + num2 = 16.0

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 20

Enter Second Number 8

Enter your choice : 2

num1 - num 2 = 12.0

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 22

Enter Second Number 3

Enter your choice : 3

num1 \* num 2 = 66.0

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 24

Enter Second Number 3

Enter your choice : 4

num1 / num 2 = 8.0

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 5

Enter Second Number 4

Enter your choice : 5

Program Finished Successfully

#Question 5

## Greatest of Three Numbers

a = 10

b=89

c=67

mx = -99999999999

if (a>b) and (a>c):

  mx = a

elif b>a and b>c:

  mx = b

else:

  mx  = c

print(mx)

#OUTPUT#

89

#Question 6

# Swap two number

a=10

b=5

print("Before Swap : b=",b,"  a=",a)

a,b=b,a

print("After Swap : ","b=",b," a=",a)

#OUTPUT#

Before Swap : b= 5 a= 10

After Swap : b= 10 a= 5

#Question 7

# ODD or Even

def odd\_even(num):

  if (num%2==0):

    print("Even")

  else:

    print("Odd")

num=int(input("Enter number to be checked : "))

odd\_even(num)

#OUTPUT#

Enter number to be checked : 32

Even

#Question 8

## Roots of Quadratic Equation

import cmath

a=int(input("Enter coefficient of x^2 : "))

b=int(input("Enter coefficient of x : "))

c=int(input("Enter constant : "))

D=(b\*\*2)-(4\*a\*c)

x1=(-b+ (cmath.sqrt(D)))/(2\*a)

x2=(-b - (cmath.sqrt(D)))/(2\*a)

if (D<0):

  print("Roots are complex : ")

  c=complex(x1,x2)

  print(c," ", c.conjugate())

elif (D==0):

  x1=-b/(2\*a)

  print("Equal Roots are ",x1)

elif(D>0):

  print("Roots are : ",x1 ," ",x2)

#OUTPUT#

Enter coefficient of x^2 : 3

Enter coefficient of x : 5

Enter constant : 15

Roots are complex :

(1.241649932998122+1.241649932998122j) (1.241649932998122-1.241649932998122j)

**LAB 2**

#### 1

## Sum of natural numbers

sum=0

for a in range(1,11):

  sum+=a

print("Sum is ",sum)

#OUTPUT#

Sum is 55

## Question 2

## AP

first=int(input("Enter first term of series : "))

d=int(input("Enter common difference of AP : "))

n=int(input("Enter no of terms "))

sum=0

def ap(first,d,n):

  print("First",n, " terms of series is/are ")

  l=(first+(n-1)\*d)

  global sum;

  for a in range(first,l+1,d):

    print(a)

    sum+=a

ap(first,d,n)

print("Sum = ",sum)

##OUTPUT##

Enter first term of series : 5

Enter common difference of AP : 5

Enter no of terms 5

First 5 terms of series is/are

5

10

15

20

25

Sum = 75

#### Question 3

## GP

first=int(input("Enter first term of series : "))

d=int(input("Enter GP factor : "))

n=int(input("Enter no of terms : "))

sum=0

def gp(first,d,n):

  global sum

  for a in range(n):

    print(first)

    sum+=first

    first\*=d

gp(first,d,n)

print("Sum = ",sum)

#OUTPUT#

Enter first term of series : 2

Enter GP factor : 2

Enter no of terms : 5

2

4

8

16

32

Sum = 62

##Question 4

##Prime check

num=int(input("Enter number to be checked : "))

def prime(num):

  flag=0

  for a in range(2,(num//2)):

    if ((num%a)==0):

      flag=1

      break

  return flag

res=prime(num)

if (res==0):

  print("Prime")

else:

  print("Non Prime")

#OUTPUT#

Enter number to be checked : 23

Prime

##Question 5

## Factorial

num=int(input("Enter number to find factorial of : "))

def factorial(num):

  for a in range(1,num):

    num\*=a

  return num

print(factorial(num))

#OUTPUT#

Enter number to find factorial of : 10

3628800

## Question 6

num=int(input("Enter number of terms of Fibonacci Series : "))

a=0

b=1

cnt=0

while (cnt<num):

  print(a)

  b,a=a+b,b

  cnt+=1

#OUTPUT#

Enter number of terms of Fibonacci Series : 5

0

1

1

2

3

## Question 7

x1=3

y1=4

z1=complex(x1,y1)

z2=complex(x1+1,y1+1) ## z2= 4i +5j

print("z2+z1=",z2+z1)

print("z1\*z2=",z1\*z2)

print("z1-z2=",z1-z2)

#OUTPUT#

z2+z1= (7+9j)

z1\*z2= (-8+31j)

z1-z2= (-1-1j)

## Question 8

import cmath

c=complex(input("Enter a complex number"))

print("Complex number entered is", c)

print("Magnitude =" ,abs(c))

print("Phase Angle(in degrees)=",(180/cmath.pi)\*cmath.phase(c))

#OUTPUT#

Enter a complex number3+4j

Complex number entered is (3+4j)

Magnitude = 5.0

Phase Angle= 53.13010235415598

## Question 9

import numpy as np

arr=np.arange(1,11)

sum=0

print(arr)

for a in range(0,10):

  sum+=arr[a]

print("Sum is =",sum)

#OUTPUT#

[ 1 2 3 4 5 6 7 8 9 10]

Sum is = 55

##Question 10

s = input("Enter String : ")

lst=list(s)

l=len(lst)

for a in range(1,l+1):

  print(lst[-a], end='')

#OUTPUT#

Enter String : asdf

Fdsa

##Question 11

s = input("Enter String : ")

lst=list(s)

l=len(lst)

cnt=-1

elem\_count=0

f=input("Enter element to be searched : ")

for a in s:

  cnt+=1

  if a==f:

    elem\_count+=1

    if elem\_count==1:

      print("First Occurence  at index ",cnt)

  elif(cnt==l) :

    print("Element not found")

print("Total Occurences = ",elem\_count)

#OUTPUT#

Enter String : aaaasss

Enter element to be searched : s

First Occurence , Element found at index 4

Total Occurences = 3

##Question 12

# Smallest and largest element in array

import numpy as np

arr=np.random.randint(100,size=10)

print(arr)

print("Max : " ,max(arr))

print("Min : ",min(arr))

#OUTPUT#

[37 79 87 29 47 45 74 17 20 4]

Max : 87

Min : 4

##Question 13

num=int(input("Enter number of terms of Fibonacci Series : "))

a=0

b=1

cnt=0

arr=[]

while (cnt<num):

  arr.append(a)

  b,a=a+b,b

  cnt+=1

print(arr)

#OUTPUT#

Enter number of terms of Fibonacci Series : 5

[0, 1, 1, 2, 3]

**LAB 3**

#Question 1

#To search a given element in the list and also specify the number of times the element has appeared in the list.

ls=eval(input("Enter list "))

elem=int(input("Enter element to be searched : "))

elem\_cnt=0

first=0

for e in range(0,len(ls)):

  if ls[e]==elem:

    elem\_cnt+=1

    if elem\_cnt==1:

      first=e

if elem\_cnt==0:f

  print("Element not found")

else:

  print(f"Total {elem\_cnt} occurences of element {elem}")

  print(f"First occurence at index {first}")

#OUTPUT#

Enter list [1,3,4,3,2,3,6]

Enter element to be searched : 3

Total 3 occurences of element 3

First occurence at index 1

#Question 2

#Create a list of ten integer elements. Remove all the odd elements from the list and retain only the even elements. Display the modified list.

import numpy as np

arr=np.random.randint(0,100,size=10)

arr=list(arr)

print("Original Array: " ,arr)

ln=len(arr)

new=[]

for a in range(0,ln):

  if (((arr[a])%2)==0):

    new.append(arr[a])

print("New Array is : ",new)

#OUTPUT#

Original Array: [55, 79, 85, 7, 77, 32, 72, 1, 9, 61]

New Array is : [32, 72]

#Question 3

#Create a list 'COLORS' which contains names of various colors such as "red", "blue" and "green".

#Also display the following MENU option to the user to perform any one of the following operations:

                  # 1. ADD a new color

                  # 2. REMOVE a color

                  # 3. SEARCH for a given color

                  # 4. SORT the colors in alphabetical order

lst=['red','yellow','green','cyan','purple','black']

def menu():

    print("\n-----------------------------------")

    print("| 1: Add Color  , 2: Remove       |\n| 3: Search  , 4:Sort  , 5:Break  |")

    print("-----------------------------------\n")

    op=int(input("Enter your choice : "))

    if (op==1):

        col=input("color : ")

        lst.append(col)

        print("Updated list looks like ",lst)

    elif (op==2):

        col=input("color : ")

        lst.remove(col)

        print("Updated list looks like ",lst)

    elif (op==3):

        col=input("color : ")

        for a in lst:

            if a==col:

                print("Found at index ",lst.index(a))

        print("Updated list looks like ",lst)

    elif (op==4):

        lst.sort()

        print("Sorted list looks like ",lst)

    elif (op==5):

      return

    menu()

print("Original list ",lst)

menu()

print(lst)

#OUTPUT#

Enter list of students

Enter name of student 1 :Zenith

Enter name of student 2 :Aman

Enter name of student 3 :Nelson

Enter name of student 4 :Kennedy

Enter name of student 5 :Jawahar

Enter name of student 6 :Mohit

Enter name of student 7 :Jack

Enter name of student 8 :Abhimanyu

Enter name of student 9 :Doraemon

Enter name of student 10 :Billy

Original List ['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy']

Sorted List ['Abhimanyu', 'Aman', 'Billy', 'Doraemon', 'Jack', 'Jawahar', 'Kennedy', 'Mohit', 'Nelson', 'Zenith']

#Question 5

#Write a program to accept a string from the user and check whether it is a palindrome or not.

string=input("Enter string ")

def p\_check(string):

  string=string.lower() #I have converted full string to lower case to remove errors due to different case

  for a in range(0,len(string)//2):

    if (string[a]==string[len(string)-a-1]):

      continue

    else:

      return "Not a Pallindrome"

  return "Is a Pallindrome"

print(string ,p\_check(string))

#OUTPUT#

Enter string Malayalam

Malayalam Is a Pallindrome

#Question 6

#Write a program to store marks of 10 students in a list. Determine the  number of students scoring,  ≤75% marks, between 75 to 85% marks and ≥ 85% marks.

lt75=0

bw75\_85=0

mt85=0

marks=[]

for a in range(10):

  mark=int(input(f"Enter marks of student {a+1} :"))

  arr.append(mark)

  if (mark<=75):

    lt75+=1

  elif(mark>75 and mark<85):

    bw75\_85+=1

  elif (mark>=85):

    mt85+=1

print("Student scoring less than or equal to 75 : ",lt75)

print("Student scoring less than 85 and more than 75 : ",bw75\_85)

print("Student scoring more than or equal to 85 : ",mt85)

#OUTPUT#

Enter marks of student 1 :34

Enter marks of student 2 :75

Enter marks of student 3 :77

Enter marks of student 4 :72

Enter marks of student 5 :55

Enter marks of student 6 :85

Enter marks of student 7 :99

Enter marks of student 8 :75

Enter marks of student 9 :83

Enter marks of student 10 :91

Student scoring less than or equal to 75 : 5

Student scoring less than 85 and more than 75 : 2

Student scoring more than or equal to 85 : 3

# Question 7

#Write a program to create a list containing names of ten students. Display a menu option to the

#user to 1.ADD a name in the list 2. Remove a name from the list 3. Sort the list in alphabetical order and 4. Search for a name in the list

lst=['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy']

def menu():

    print("\n-----------------------------------")

    print("| 1: Add Name , 2: Remove Name    |\n| 3: Sort  , 4:Search  , 5:Break  |")

    print("-----------------------------------\n")

    op=int(input("Enter your choice : "))

    if (op==1):

        col=input("Name to be added : ")

        lst.append(col)

        print("Updated list looks like ",lst)

    elif (op==2):

        col=input("Name to be removed : ")

        lst.remove(col)

        print("Updated list looks like ",lst)

    elif (op==4):

        col=input("Enter name to be searched : ")

        for a in lst:

            if a==col:

                print("Found at index ",lst.index(a))

    elif (op==3):

        lst.sort()

        print("Sorted list looks like ",lst)

    elif (op==5):

      return

    menu()

print("Original list ",lst)

menu()

print(lst)

#OUTPUT#

Original list ['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy']

-----------------------------------

| 1: Add Name , 2: Remove Name |

| 3: Sort , 4:Search , 5:Break |

-----------------------------------

Enter your choice : 1

Name to be added : Zatch

Updated list looks like ['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy', 'Zatch']

-----------------------------------

| 1: Add Name , 2: Remove Name |

| 3: Sort , 4:Search , 5:Break |

-----------------------------------

Enter your choice : 2

Name to be removed : Jawahar

Updated list looks like ['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy', 'Zatch']

-----------------------------------

| 1: Add Name , 2: Remove Name |

| 3: Sort , 4:Search , 5:Break |

-----------------------------------

Enter your choice : 3

Sorted list looks like ['Abhimanyu', 'Aman', 'Billy', 'Doraemon', 'Jack', 'Kennedy', 'Mohit', 'Nelson', 'Zatch', 'Zenith']

-----------------------------------

| 1: Add Name , 2: Remove Name |

| 3: Sort , 4:Search , 5:Break |

-----------------------------------

Enter your choice : 4

Enter name to be searched : Doraemon

Found at index 3

-----------------------------------

| 1: Add Name , 2: Remove Name |

| 3: Sort , 4:Search , 5:Break |

-----------------------------------

Enter your choice : 5

['Abhimanyu', 'Aman', 'Billy', 'Doraemon', 'Jack', 'Kennedy', 'Mohit', 'Nelson', 'Zatch', 'Zenith']

**LAB 4**

# Question 1

#Calculator

lst=[]

def add(num1,num2):

  return num1+num2

def multiply(num1,num2):

  return num1\*num2

def subtraction(num1,num2):

  return num1-num2

def divide(num1,num2):

  return num1/num2

def menu():

    print("\n-----------------------------------")

    print("| 1: ADD         2:Subtract       |\n|3: MULTIPLY , 4:DIVIDE , 5:Break |")

    print("-----------------------------------\n")

    num1=float(input("Enter First Number "))

    num2=float(input("Enter Second Number "))

    op=int(input("Enter your choice : "))

    if (op==1):

        print("num1 + num2 = ",add(num1,num2))

    elif (op==2):

      print("num1 - num 2 = ",subtraction(num1,num2))

    elif (op==3):

      print("num1 \* num 2 = ",multiply(num1,num2))

    elif (op==4):

      print("num1 / num 2 = ",divide(num1,num2))

    elif (op==5):

      return

    else:

      print("Wrong Input , Try again")

    menu()

menu()

print("Program Finished Successfully ")

#OUTPUT#

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 34

Enter Second Number 2

Enter your choice : 4

num1 / num 2 = 17.0

-----------------------------------

| 1: ADD 2:Subtract |

|3: MULTIPLY , 4:DIVIDE , 5:Break |

-----------------------------------

Enter First Number 0

Enter Second Number 0

Enter your choice : 5

Program Finished Successfully

#Question 2

#To create a list containing names of ten students. Display a menu option to the user to 1. ADD a name in the list, 2. Remove a name from the list,

# 3. sort the list in alphabetical order and 4. search for a name in the list. (create a separate function to perform all these operations)

arr=['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy']

print("Original Array : ",arr)

def add(name):

  arr.append(name)

def remove(name):

  arr.remove(name)

def sort():

  arr.sort()

def search(name):

  for a in arr:

            if a==name:

                print("Found at index ",arr.index(a))

                return arr.index(a)

  print("Not found\n")

def menu():

    print("\n-----------------------------------")

    print("| 1: ADD STUDENT 2:REMOVE STUDENT |\n|3: SORT, 4:Search , 5:Stop , 6: Print array |")

    print("-----------------------------------\n")

    op=int(input("Enter your choice : "))

    if (op==1):

        name=input("Enter Name : ")

        add(name)

        print("Name added Successfully\n")

    elif (op==2):

      name=input("Enter Name : ")

      remove(name)

      print("Name removed successfully\n")

    elif (op==3):

      sort()

      print("Array Sorted Successfully\n")

    elif (op==4):

      name=input("Enter Name : ")

      search(name)

    elif (op==5):

      return

    elif (op==6):

      print(arr)

    else:

      print("Wrong Input , Try again")

    menu()

menu()

print("Program Finished Successfully ")

#OUTPUT#

Original Array : ['Zenith', 'Aman', 'Nelson', 'Kennedy', 'Jawahar', 'Mohit', 'Jack', 'Abhimanyu', 'Doraemon', 'Billy']

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 1

Enter Name : Test

Name added Successfully

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 2

Enter Name : Jack

Name removed successfully

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 3

Array Sorted Successfully

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 4

Enter Name : Aman

Found at index 1

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 6

['Abhimanyu', 'Aman', 'Billy', 'Doraemon', 'Jawahar', 'Kennedy', 'Mohit', 'Nelson', 'Test', 'Zenith']

-----------------------------------

| 1: ADD STUDENT 2:REMOVE STUDENT |

|3: SORT, 4:Search , 5:Stop , 6: Print array |

-----------------------------------

Enter your choice : 5

Program Finished Successfully

#Question 3

#To Create three sets (SET1, SET2 and SET3)

SET1={'Abhimanyu', 'Aman', 'Billy', 'Doraemon', 'Jawahar', 'Kennedy', 'Mohit', 'Nelson', 'Test', 'Zenith'}

SET2={'Kennedy', 'Mohit', 'Nelson', 'Test', 'Zenith'} #students scoring more than 75% in English

SET3={'Doraemon', 'Jawahar', 'Kennedy', 'Mohit'} # students scoring more than 75% in Maths

print(SET1)

print("Students scoring more than 75% in English ",SET1)

print("Students scoring more than 75% in Maths ",SET2)

print("Students scoring more than 75% in both English and Maths.",(SET2 & SET3))

print("Students scoring more than 75% in English but not in Maths.",(SET2 - SET3))

print("Students scoring more than 75% in Maths but not in English. ",(SET3-SET2))

print("Students who were not able to scoring more than 75% in Maths as well as English ",(SET1 - (SET2 | SET3)),end="\n\n")

#OUTPUT#

{'Jawahar', 'Test', 'Billy', 'Aman', 'Mohit', 'Kennedy', 'Abhimanyu', 'Zenith', 'Doraemon', 'Nelson'}

Students scoring more than 75% in English {'Jawahar', 'Test', 'Billy', 'Aman', 'Mohit', 'Kennedy', 'Abhimanyu', 'Zenith', 'Doraemon', 'Nelson'}

Students scoring more than 75% in Maths {'Test', 'Zenith', 'Mohit', 'Kennedy', 'Nelson'}

Students scoring more than 75% in both English and Maths. {'Mohit', 'Kennedy'}

Students scoring more than 75% in English but not in Maths. {'Zenith', 'Test', 'Nelson'}

Students scoring more than 75% in Maths but not in English. {'Jawahar', 'Doraemon'}

Students who were not able to scoring more than 75% in Maths as well as English {'Aman', 'Abhimanyu', 'Billy'}

**LAB 5**

## 1

Tree=['A','B','C','D','E','F','G']

Queue=[]

root=Tree[0]

initial=root

goal=input

Queue.append(root)

s=0

if Queue[0]==goal:

  print("Goal Node found Search Successful!!!")

  print("Goal Node is",Queue[0])

  s=1

if s==0:

  print(Queue[0])

  Queue.pop(0)

  i=1

  while i<=6:

   Queue.append(Tree[i])

   i=i+1

   Queue.append(Tree[i])

   i=i+1

   print(Queue[0])

   if Queue[0]==goal:

    print("Search Successful! Goal node found!!!!")

    s=1

    break

   Queue.pop(0)

if s==0:

     while Queue!=[]:

       print(Queue[0])

       if Queue[0]==goal:

        print("Search complete")

        s=1

        break

       Queue.pop(0)

if s==0:

  print("Search Incomplete!! Goal state not found")

A

B

C

D

E

F

G

Search Incomplete!! Goal state not found

##2

Tree=['A','B','C','D','E','F','G']

Queue=[]

root=Tree[0]

initial=root

goal='E'

Queue.append(root)

s=0

if Queue[0]==goal:

  print("Goal Node found Search Successful!!!")

  print("Goal Node is",Queue[0])

  s=1

if s==0:

  print(Queue[0])

  Queue.pop(0)

  i=1

  while i<=6:

   Queue.append(Tree[i])

   i=i+1

   Queue.append(Tree[i])

   i=i+1

   print(Queue[0])

   if Queue[0]==goal:

    print("Search Successful! Goal node found!!!!")

    s=1

    break

   Queue.pop(0)

if s==0:

     while Queue!=[]:

       print(Queue[0])

       if Queue[0]==goal:

        print("Search complete")

        s=1

        break

       Queue.pop(0)

if s==0:

  print("Search Incomplete!! Goal state not found")

#OUTPUT#

A

B

C

D

E

Search complete

#BFS FROM RIGHT HAND SIDE

##3

Tree1=['A','C','B','E','D','G','F']

print(Tree1)

Queue=[]

root=Tree1[0]

initial=root

goal=input("Enter The Goal state which you want to find from the above tree : ")

Queue.append(root)

s=0

if Queue[0]==goal:

  print("Goal Node found Search Successful!!!")

  print("Goal Node is",Queue[0])

  s=1

if s==0:

  print(Queue[0])

  Queue.pop(0)

  i=1

  while (i<6):

   Queue.append(Tree1[i])

   i=i+1

   Queue.append(Tree1[i])

   i=i+1

   print(Queue[0])

   if Queue[0]==goal:

    print("Search Successful! Goal node found!!!!")

    s=1

    break

   Queue.pop(0)

if s==0:

     while Queue!=[]:

       print(Queue[0])

       if Queue[0]==goal:

        print("Search complete!! GOAL NODE FOUND!!!")

        s=1

        break

       Queue.pop(0)

if s==0:

  print("Search Incomplete!! Goal state", goal, "not found")

#OUTPUT#

['A', 'C', 'B', 'E', 'D', 'G', 'F']

Enter The Goal state which you want to find from the above tree : F

A

C

B

E

D

G

F

Search complete!! GOAL NODE FOUND!!!

##4

Tree1=[]

d = int(input("Please enter the depth of the tree"))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

Tree1.append(a)

i=i+1

while i<N:

  print("Enter left child of element of", Tree1[j])

  print("If there is no left child , enter 'null'")

  a=input()

  Tree1.append(a)

  i=i+1

  print("Enter right child of element ", Tree1[j])

  print("If there is no right child , enter 'null'")

  a=input()

  Tree1.append(a)

  i=i+1

  j=j+1

print(Tree1)

c=0

for l in range(0,d+1):

  space =d-l

  for m in range(0,space):

      print(" ", end=" ")

  for n in range(0, 2\*\*l):

    print(Tree[c]," ",end = " ")

    c=c+1

  print("\n")

  Queue=[]

root=Tree1[0]

initial=root

goal=input("Enter The Goal state which you want to find from the above tree : ")

Queue.append(root)

s=0

if Queue[0]==goal:

  print("Goal Node found Search Successful!!!")

  print("Goal Node is",Queue[0])

  s=1

if s==0:

  print(Queue[0])

  Queue.pop(0)

  i=1

  while (i<N):

   Queue.append(Tree1[i])

   i=i+1

   Queue.append(Tree1[i])

   i=i+1

   print(Queue[0])

   if Queue[0]==goal:

    print("Search Successful! Goal node found!!!!")

    s=1

    break

   Queue.pop(0)

if s==0:

     while Queue!=[]:

       print(Queue[0])

       if Queue[0]==goal:

        print("Search complete!! GOAL NODE FOUND!!!")

        s=1

        break

       Queue.pop(0)

if s==0:

  print("Search Incomplete!! Goal state", goal, "not found")

#OUTPUT#

Please enter the depth of the tree2

7

Enter element 0 (root node)

A

Enter left child of element of A

If there is no left child , enter 'null'

B

Enter right child of element A

If there is no right child , enter 'null'

C

Enter left child of element of B

If there is no left child , enter 'null'

D

Enter right child of element B

If there is no right child , enter 'null'

E

Enter left child of element of C

If there is no left child , enter 'null'

F

Enter right child of element C

If there is no right child , enter 'null'

G

['A', 'B', 'C', 'D', 'E', 'F', 'G']

A

B C

D E F G

Enter The Goal state which you want to find from the above tree : E

A

B

C

D

E

Search complete!! GOAL NODE FOUND!!!

##5

Tree1=[]

d = int(input("Please enter the depth of the tree"))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

Tree1.append(a)

i=i+1

while i<N:

  print("Enter left child of element of", Tree1[j])

  print("If there is no left child , enter '\*'")

  a=input()

  Tree1.append(a)

  i=i+1

  print("Enter right child of element ", Tree1[j])

  print("If there is no right child , enter '\*'")

  a=input()

  Tree1.append(a)

  i=i+1

  j=j+1

print(Tree1)

c=0

for l in range(0,d+1):

  space =d-l

  for m in range(0,space):

      print(" ", end=" ")

  for n in range(0, 2\*\*l):

    print(Tree1[c]," ",end = " ")

    c=c+1

  print("\n")

  Queue=[]

root=Tree1[0]

initial=root

goal=input("Enter The Goal state which you want to find from the above tree : ")

Queue.append(root)

s=0

if Queue[0]==goal:

  print("Goal Node found Search Successful!!!")

  print("Goal Node is",Queue[0])

  s=1

if s==0:

  print(Queue[0])

  Queue.pop(0)

  i=1

  while (i<N):

   Queue.append(Tree1[i])

   i=i+1

   Queue.append(Tree1[i])

   i=i+1

   print(Queue[0])

   if Queue[0]==goal:

    print("Search Successful! Goal node found!!!!")

    s=1

    break

   Queue.pop(0)

if s==0:

     while Queue!=[]:

       print(Queue[0])

       if Queue[0]==goal:

        print("Search complete!! GOAL NODE FOUND!!!")

        s=1

        break

       Queue.pop(0)

if s==0:

  print("Search Incomplete!! Goal state", goal, "not found")

#OUTPUT#

If there is no left child , enter '\*'

\*

Enter right child of element \*

If there is no right child , enter '\*'

\*

Enter left child of element of I

If there is no left child , enter '\*'

\*

Enter right child of element I

If there is no right child , enter '\*'

\*

Enter left child of element of \*

If there is no left child , enter '\*'

\*

Enter right child of element \*

If there is no right child , enter '\*'

\*

Enter left child of element of \*

If there is no left child , enter '\*'

\*

Enter right child of element \*

If there is no right child , enter '\*'

\*

Enter left child of element of \*

If there is no left child , enter '\*'

\*

Enter right child of element \*

If there is no right child , enter '\*'

\*

['S', 'A', 'B', 'C', 'D', 'G', 'H', 'E', 'F', '\*', '\*', 'I', '\*', '\*', '\*', 'K', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*']

S

A B

C D G H

E F \* \* I \* \* \*

K \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

Enter The Goal state which you want to find from the above tree : K

S

A

B

C

D

G

H

E

F

\*

\*

I

\*

\*

\*

K

Search Successful! Goal node found!!!!

**LAB 6**

# Program 1 (SEARCH TREE 1)

#  To determine the LEFTMOST STRAIGHT LINE PATH to move from Node A to Node F

Tree=['A', 'C', 'B', 'E', 'D','G', 'F']

d=2

STACK= []

root= Tree[0]

initial= root

goal='F'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

#OUTPUT#

A

B

D

E

C

F

Search Successful! Goal Node Found!

# -Program 2 (SEARCH TREE 1)

#  To determine the RIGHTMOST STRAIGHT LINE PATH to move from Node A to Node E

Tree=['A','B','C','F','G','D','E']

d=2

STACK= []

root= Tree[0]

initial= root

goal='E'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    # print("\n",STACK,"\n")

    print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

\*OUTPUT\*

A

C

G

F

B

E

Search Successful! Goal Node Found!

# Program 3 (Search Tree 2)

#  To determine the LEFTMOST STRAIGHT LINE PATH to move from Node S to Node G

Tree=['S','H','A','C','B','J','G']

d=2

STACK= []

root= Tree[0]

initial= root

goal='G'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    # print("\n",STACK,"\n")

    print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

\*OUTPUT\*

S

A

B

C

H

G

Search Successful! Goal Node Found!

# Program 4(SEARCH TREE 2)

#  To determine the RIGHTMOST STRAIGHT LINE PATH to move from Node S to Node H

Tree= ['s','a','h','g','j','b','c']

d=2

STACK= []

root= Tree[0]

initial= root

goal='h'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    # print("\n",STACK,"\n")

    print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

\*OUTPUT\*

s

h

Search Successful! Goal Node Found!

# Program 5 (SEARCH TREE 3)

#  To determine the LEFTMOST STRAIGHT LINE PATH  to move from Node A to Node F

Tree=tree=['a','c','b','f','e','f','null']

d=2

STACK= []

root= Tree[0]

initial= root

goal='f'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    # print("\n",STACK,"\n")

    if STACK[TOP]!='null':

      print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

\*OUTPUT\*

a

b

e

f

Search Successful! Goal Node Found!

# -Program 6  (SEARCH TREE 2)

#  To determine the RIGHTMOST STRAIGHT LINE PATH to move from Node A to Node B

Tree=['a','b','c','f','null','e','f']

d=2

STACK= []

root= Tree[0]

initial= root

goal='b'

STACK.append(root)

s=0

if STACK[0]==goal:

  print("Goal Node Found! Search Successful!")

  print(STACK[0],"is the goal node")

  s=1

if s==0:

  print(STACK[0])

  STACK.pop()

i=1

j=1

while s==0 and i<=6:

  if j<=d:

    STACK.append(Tree[i])

    i=i+1

    STACK.append(Tree[i])

    i=i+1

    TOP=len(STACK)-1

    # print("\n",STACK,"\n")

    if STACK[TOP]!='null':

      print(STACK[TOP])

    if STACK[TOP]==goal:

      print("Search Successful! Goal Node Found!")

      s=1

      break

    STACK.pop()

    j=j+1

  if j>d:

    while STACK!=[]:

      TOP=len(STACK)-1

      print(STACK[TOP])

      if STACK[TOP]==goal:

        print("Search Successful! Goal Node Found!")

        s=1

        break

      STACK.pop()

    j=j-1

if s==0:

  print("Goal State not Found! Search Unsuccessful")

\*OUTPUT\*

a

c

f

b

Search Successful! Goal Node Found!

**LAB 7**

#1

OPEN=[]

CLOSE=[]

s=0

final='G'

print("Enter the elements of the tree\n")

print('Enter First element (root node)\n')

root=input()

OPEN.append(root)

if root==final:

  print("GOal Node Found")

  print("The path is ",root)

  s=1

else:

  OPEN.pop()

  CLOSE.append(root)

currentnode=root

ans='y'

while s==0 and (ans in ['y','Y']):

  ans=input("Do you want to add more?<y/n> ")

  print("Enter Left child of node ",currentnode)

  left=input()

  print("Enter Heuristic value of ",left)

  HL=int(input())

  print("Enter Right child of node ",currentnode)

  right=input()

  print("Enter Heuristic value of ",right)

  HR=int(input())

  if HL<HR:

    CLOSE.append(left)

    OPEN.append(right)

    currentnode=left

    if left==final:

      print("Goal Node Found\nThe Path is ",CLOSE)

      s=1

      break

  elif HR<=HL:

    CLOSE.append(right)

    OPEN.append(left)

    currentnode=right

    if right==final:

      print("Goal Node Found\nThe Path is ",CLOSE)

      s=1

      break

 \*OUTPUT\*

Enter the elements of the tree

Enter First element (root node)

S

Do you want to add more?<y/n> y

Enter Left child of node S

A

Enter Heuristic value of A

3

Enter Right child of node S

G

Enter Heuristic value of G

0

Goal Node Found

The Path is ['S', 'G']

#2 Greedy First Search

open=[]

close=[]

s=0

final='G'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  print("Enter the heuristic value of", left )

  hl= int(input())

  print("Enter the right child of", currentnode)

  right=input()

  print("Enter the heuristic value of", right )

  hr= int(input())

  if hl<hr:

    close.append(left)

    open.append(right)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif (hr<=hl):

      close.append(right)

      open.append(left)

      currentnode=right

      if right==final:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

\*OUTPUT\*

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the left child of S

A

Enter the heuristic value of A

3

Enter the right child of S

G

Enter the heuristic value of G

0

Goal Node found!

The path is:

['S', 'G']

#3

open=[]

close=[]

s=0

final='J'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

  if hl<hr and hl<hm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif hm<hr and hm<hl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if mid==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif (hr<=hl and hr<=hm):

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if right==final:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

\*OUTPUT\*

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the left child of S

A

Enter the heuristic value of A

6

Enter the middle child of S

Enter the right child of S

b

Enter the heuristic value of b

2

Open List: ['A']

Closed List: ['S', 'b']

Do you want to continue? <y/n>

y

Enter the left child of b

C

Enter the heuristic value of C

1

Enter the middle child of b

A

Enter the heuristic value of A

6

Enter the right child of b

Open List: ['A', 'A']

Closed List: ['S', 'b', 'C']

Do you want to continue? <y/n>

y

Enter the left child of C

A

Enter the heuristic value of A

6

Enter the middle child of C

G

Enter the heuristic value of G

0

Enter the right child of C

Open List: ['A', 'A', 'A']

Closed List: ['S', 'b', 'C', 'G']

Do you want to continue? <y/n>

n

Goal node not found!

# Tree 4

open=[]

close=[]

s=0

final='G'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

  if hl<hr and hl<hm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif hm<hr and hm<hl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if mid==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif (hr<=hl and hr<=hm):

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if right==final:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the left child of S

A

Enter the heuristic value of A

6

Enter the middle child of S

B

Enter the heuristic value of B

2

Enter the right child of S

Open List: ['A']

Closed List: ['S', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

A

Enter the heuristic value of A

6

Enter the middle child of B

C

Enter the heuristic value of C

1

Enter the right child of B

Open List: ['A', 'A']

Closed List: ['S', 'B', 'C']

Do you want to continue? <y/n>

y

Enter the left child of C

A

Enter the heuristic value of A

6

Enter the middle child of C

G

Enter the heuristic value of G

0

Enter the right child of C

Goal Node found!

The path is:

['S', 'B', 'C', 'G']

#5 Greedy First

OPEN=[]

CLOSE=[]

s=0

final='G'

print("Enter the elements of the tree\n")

print('Enter First element (root node)\n')

root=input()

OPEN.append(root)

if root==final:

  print("GOal Node Found")

  print("The path is ",root)

  s=1

else:

  OPEN.pop()

  CLOSE.append(root)

currentnode=root

ans='y'

while s==0 :

  ans=input("Do you want to add more?<y/n> ")

  if ans not in ['y','Y']:

    break

  print("Enter Left child of node ",currentnode)

  left=input()

  print("Enter Heuristic value of ",left)

  HL=int(input())

  print("Enter Right child of node ",currentnode)

  right=input()

  print("Enter Heuristic value of ",right)

  HR=int(input())

  if HL<HR:

    CLOSE.append(left)

    OPEN.append(right)

    currentnode=left

    if left==final:

      print("Goal Node Found\nThe Path is ",CLOSE)

      s=1

      break

  elif HR<=HL:

    CLOSE.append(right)

    OPEN.append(left)

    currentnode=right

    if right==final:

      print("Goal Node Found\nThe Path is ",CLOSE)

      s=1

      break

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree

Enter First element (root node)

S

Do you want to add more?<y/n> y

Enter Left child of node S

A

Enter Heuristic value of A

5

Enter Right child of node S

B

Enter Heuristic value of B

4

Do you want to add more?<y/n> y

Enter Left child of node B

C

Enter Heuristic value of C

5

Enter Right child of node B

E

Enter Heuristic value of E

2

Do you want to add more?<y/n> y

Enter Left child of node E

D

Enter Heuristic value of D

2

Enter Right child of node E

F

Enter Heuristic value of F

1

Do you want to add more?<y/n> n

Goal node not found!

#Program 6 for Greedy Best First Search

open=[]

close=[]

s=0

final='g'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

  if hl<hr and hl<hm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif hm<hr and hm<hl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if mid==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif (hr<=hl and hr<=hm):

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if right==final:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

c

Enter the left child of c

Enter the middle child of c

b

Enter the heuristic value of b

6

Enter the right child of c

Open List: []

Closed List: ['c', 'b']

Do you want to continue? <y/n>

y

Enter the left child of b

a

Enter the heuristic value of a

11

Enter the middle child of b

g

Enter the heuristic value of g

0

Enter the right child of b

Goal Node found!

The path is:

['c', 'b', 'g']

**LAB 8**

#Program 1 for A Star Search

open=[]

close=[]

s=0

final='G'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr:

    close.append(left)

    if(right!=""):

      open.append(right)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

    close.append(right)

    if(left!=""):

      open.append(left)

    currentnode=right

    if right==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the left child of S

A

Enter the heuristic value of A

12

Enter the path cost of A

3

Enter the right child of S

B

Enter the heuristic value of B

4

Enter the path cost of B

2

Open List: ['A']

Closed List: ['S', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

E

Enter the heuristic value of E

8

Enter the path cost of E

3

Enter the right child of B

F

Enter the heuristic value of F

2

Enter the path cost of F

1

Open List: ['A', 'E']

Closed List: ['S', 'B', 'F']

Do you want to continue? <y/n>

y

Enter the left child of F

I

Enter the heuristic value of I

9

Enter the path cost of I

2

Enter the right child of F

G

Enter the heuristic value of G

0

Enter the path cost of G

3

Goal Node found!

The path is:

['S', 'B', 'F', 'G']

#Program 2 for A Star Search

open=[]

close=[]

s=0

hv=0

print("Enter the elements of the tree: \n")

print("Enter the first element(root node): \n")

root=input()

print("Enter the heuristic value of the first element: \n")

h=input()

open.append(root)

if h==hv:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr:

    close.append(left)

    if(right!=""):

      open.append(right)

    currentnode=left

    if hl==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

    close.append(right)

    if(left!=""):

      open.append(left)

    currentnode=right

    if hr==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node):

S

Enter the heuristic value of the first element:

13

Enter the left child of S

A

Enter the heuristic value of A

12

Enter the path cost of A

3

Enter the right child of S

B

Enter the heuristic value of B

4

Enter the path cost of B

2

Open List: ['A']

Closed List: ['S', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

E

Enter the heuristic value of E

8

Enter the path cost of E

1

Enter the right child of B

F

Enter the heuristic value of F

2

Enter the path cost of F

3

Open List: ['A', 'E']

Closed List: ['S', 'B', 'F']

Do you want to continue? <y/n>

y

Enter the left child of F

I

Enter the heuristic value of I

0

Enter the path cost of I

5

Enter the right child of F

G

Enter the heuristic value of G

0

Enter the path cost of G

3

Goal Node found!

The path is:

['S', 'B', 'F', 'G']

#Program 3 for A Star Search

open=[]

close=[]

s=0

hv=0

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

print("Enter the heuristic value of the first element: \n")

h=input()

open.append(root)

if h==hv:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

tr=999999

tl=999999

tm=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

    print("Enter the path cost of", mid )

    pcm= int(input())

    tm=hm+pcm

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr and tl<tm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if hl==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif tm<tr and tm<tl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if hm==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if hr==hv:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the heuristic value of the first element:

7

Enter the left child of S

B

Enter the heuristic value of B

2

Enter the path cost of B

6

Enter the middle child of S

Enter the right child of S

A

Enter the heuristic value of A

6

Enter the path cost of A

2

Open List: ['B']

Closed List: ['S', 'A']

Do you want to continue? <y/n>

y

Enter the left child of A

B

Enter the heuristic value of B

2

Enter the path cost of B

2

Enter the middle child of A

C

Enter the heuristic value of C

1

Enter the path cost of C

4

Enter the right child of A

G

Enter the heuristic value of G

0

Enter the path cost of G

8

Open List: ['B', 'G', 'C']

Closed List: ['S', 'A', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

C

Enter the heuristic value of C

1

Enter the path cost of C

1

Enter the middle child of B

Enter the right child of B

Open List: ['B', 'G', 'C']

Closed List: ['S', 'A', 'B', 'C']

Do you want to continue? <y/n>

y

Enter the left child of C

G

Enter the heuristic value of G

0

Enter the path cost of G

3

Enter the middle child of C

Enter the right child of C

Goal Node found!

The path is:

['S', 'A', 'B', 'C', 'G']

#Program 4 for A Star Search

open=[]

close=[]

s=0

hv=0

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

print("Enter the heuristic value of the first element: \n")

h=input()

open.append(root)

if h==hv:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

tr=999999

tl=999999

tm=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

    print("Enter the path cost of", mid )

    pcm= int(input())

    tm=hm+pcm

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr and tl<tm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if hl==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif tm<tr and tm<tl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if hm==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if hr==hv:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

A

Enter the heuristic value of the first element:

10

Enter the left child of A

B

Enter the heuristic value of B

8

Enter the path cost of B

6

Enter the middle child of A

Enter the right child of A

F

Enter the heuristic value of F

6

Enter the path cost of F

3

Open List: ['B']

Closed List: ['A', 'F']

Do you want to continue? <y/n>

y

Enter the left child of F

G

Enter the heuristic value of G

5

Enter the path cost of G

1

Enter the middle child of F

Enter the right child of F

H

Enter the heuristic value of H

3

Enter the path cost of H

7

Open List: ['B', 'H']

Closed List: ['A', 'F', 'G']

Do you want to continue? <y/n>

y

Enter the left child of G

Enter the middle child of G

Enter the right child of G

I

Enter the heuristic value of I

1

Enter the path cost of I

3

Open List: ['B', 'H']

Closed List: ['A', 'F', 'G', 'I']

Do you want to continue? <y/n>

y

Enter the left child of I

J

Enter the heuristic value of J

0

Enter the path cost of J

3

Enter the middle child of I

Enter the right child of I

Goal Node found!

The path is:

['A', 'F', 'G', 'I', 'J']

#Program 5 A\* Algo

open=[]

close=[]

s=0

final='G'

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

open.append(root)

if root==final:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr:

    close.append(left)

    if(right!=""):

      open.append(right)

    currentnode=left

    if left==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

    close.append(right)

    if(left!=""):

      open.append(left)

    currentnode=right

    if right==final:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

S

Enter the left child of S

A

Enter the heuristic value of A

5

Enter the path cost of A

3

Enter the right child of S

B

Enter the heuristic value of B

4

Enter the path cost of B

2

Open List: ['A']

Closed List: ['S', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

C

Enter the heuristic value of C

5

Enter the path cost of C

1

Enter the right child of B

E

Enter the heuristic value of E

2

Enter the path cost of E

4

Open List: ['A', 'C']

Closed List: ['S', 'B', 'E']

Do you want to continue? <y/n>

y

Enter the left child of E

D

Enter the heuristic value of D

2

Enter the path cost of D

3

Enter the right child of E

F

Enter the heuristic value of F

1

Enter the path cost of F

1

Open List: ['A', 'C', 'D']

Closed List: ['S', 'B', 'E', 'F']

Do you want to continue? <y/n>

y

Enter the left child of F

G

Enter the heuristic value of G

0

Enter the path cost of G

2

Enter the right child of F

Open List: ['A', 'C', 'D', 'G']

Closed List: ['S', 'B', 'E', 'F', '']

Do you want to continue? <y/n>

n

Goal node not found!

#Program 6 for A Star Search

open=[]

close=[]

s=0

hv=0

print("Enter the elements of the tree: \n")

print("Enter the first element(root node)\n")

root=input()

print("Enter the heuristic value of the first element: \n")

h=input()

open.append(root)

if h==hv:

  print("Goal Node found!")

  print("The path is: ",root)

  s=1

else:

  open.pop()

  close.append(root)

currentnode=root

ans='y'

hr=999999

hm=999999

hl=999999

tr=999999

tl=999999

tm=999999

while s==0 and (ans=='y' or ans=='Y'):

  print("Enter the left child of", currentnode)

  left=input()

  if(left!=""):

    print("Enter the heuristic value of", left )

    hl= int(input())

    print("Enter the path cost of", left )

    pcl= int(input())

    tl=hl+pcl

  print("Enter the middle child of", currentnode)

  mid=input()

  if(mid!=""):

    print("Enter the heuristic value of", mid )

    hm= int(input())

    print("Enter the path cost of", mid )

    pcm= int(input())

    tm=hm+pcm

  print("Enter the right child of", currentnode)

  right=input()

  if(right!=""):

    print("Enter the heuristic value of", right )

    hr= int(input())

    print("Enter the path cost of", right )

    pcr= int(input())

    tr=hr+pcr

  if tl<tr and tl<tm:

    close.append(left)

    if(right!=""):

      open.append(right)

    if(mid!=""):

      open.append(mid)

    currentnode=left

    if hl==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  elif tm<tr and tm<tl:

    close.append(mid)

    if(right!=""):

      open.append(right)

    if(left!=""):

      open.append(left)

    currentnode=mid

    if hm==hv:

      print("Goal Node found!")

      print("The path is: ")

      print(close)

      s=1

      break

  else:

      close.append(right)

      if(left!=""):

        open.append(left)

      if(mid!=""):

        open.append(mid)

      currentnode=right

      if hr==hv:

        print("Goal Node found!")

        print("The path is: ")

        print(close)

        s=1

        break

  print("Open List:", open)

  print("Closed List:", close)

  print("Do you want to continue? <y/n>")

  ans=input()

if s==0:

  print("Goal node not found!")''

#OUTPUT#

Enter the elements of the tree:

Enter the first element(root node)

C

Enter the heuristic value of the first element:

99

Enter the left child of C

B

Enter the heuristic value of B

6

Enter the path cost of B

1

Enter the middle child of C

Enter the right child of C

Open List: []

Closed List: ['C', 'B']

Do you want to continue? <y/n>

y

Enter the left child of B

A

Enter the heuristic value of A

11

Enter the path cost of A

2

Enter the middle child of B

Enter the right child of B

G

Enter the heuristic value of G

0

Enter the path cost of G

9

Goal Node found!

The path is:

['C', 'B', 'G']

**LAB 9**

#Program 1

#WHEN MAX PLAYER STARTS THE GAME

GT=[]

SCORE=[]

d = int(input("Please enter the depth of the tree: "))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

GT.append(a)

print("Enter The Score of Staring element")

b=input()

SCORE.append(b)

i=i+1

while i<N:

  print("Enter left child of element of", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  print("Enter right child of element ", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  j=j+1

print(GT)

print("SCORES ARE",SCORE)

MAX=1

MIN=0

MAX=1

MIN=0

currentmove=GT[0]

CurrentScore=SCORE[0]

GAME\_PATH=[]

print("STARTING STATE OF THE GAME IS ",GT[0])

LASTLEVEL=2

N=0

while N<=LASTLEVEL:

    if MAX==0 and MIN==1:

        N=N+1

        print("MOVE ",N)

        print("IT'S MIN  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]<SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=1

        MIN=0

        if N>2:

            break

    if MAX==1 and MIN==0:

        N=N+1

        print("MOVE ",N)

        print("IT'S MAX  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]>SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=0

        MIN=1

        if N>2:

            break

if MAX==1:

    print("The final score of MIN player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

else:

    print("The final score of MAX player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

#OUTPUT#

Please enter the depth of the tree: 3

15

Enter element 0 (root node)

A

Enter The Score of Staring element

4

Enter left child of element of A

B

Enter The Score of the: B

4

Enter right child of element A

C

Enter The Score of the: C

-3

Enter left child of element of B

D

Enter The Score of the: D

4

Enter right child of element B

E

Enter The Score of the: E

6

Enter left child of element of C

F

Enter The Score of the: F

-3

Enter right child of element C

G

Enter The Score of the: G

7

Enter left child of element of D

H

Enter The Score of the: H

-1

Enter right child of element D

I

Enter The Score of the: I

4

Enter left child of element of E

J

Enter The Score of the: J

2

Enter right child of element E

K

Enter The Score of the: K

6

Enter left child of element of F

L

Enter The Score of the: L

-3

Enter right child of element F

M

Enter The Score of the: M

-5

Enter left child of element of G

N

Enter The Score of the: N

0

Enter right child of element G

O

Enter The Score of the: O

7

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']

SCORES ARE ['4', '4', '-3', '4', '6', '-3', '7', '-1', '4', '2', '6', '-3', '-5', '0', '7']

STARTING STATE OF THE GAME IS A

MOVE 1

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

B

C

B SELECTED!!

MOVE 2

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

D

E

D SELECTED!!

MOVE 3

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

H

I

I SELECTED!!

The final score of MAX player is 4

The Final path is ['B', 'D', 'I']

#P-1

#WHEN MIN PLAYER STARTS THE GAME

GT=[]

SCORE=[]

d = int(input("Please enter the depth of the tree: "))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

GT.append(a)

print("Enter The Score of Staring element")

b=input()

SCORE.append(b)

i=i+1

while i<N:

  print("Enter left child of element of", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  print("Enter right child of element ", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  j=j+1

print(GT)

print("SCORES ARE",SCORE)

MAX=0

MIN=1

currentmove=GT[0]

CurrentScore=SCORE[0]

GAME\_PATH=[]

print("STARTING STATE OF THE GAME IS ",GT[0])

LASTLEVEL=2

N=0

while N<=LASTLEVEL:

    if MAX==0 and MIN==1:

        N=N+1

        print("MOVE ",N)

        print("IT'S MIN  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]<SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=1

        MIN=0

        if N>2:

            break

    if MAX==1 and MIN==0:

        N=N+1

        print("MOVE ",N)

        print("IT'S MAX  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]>SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=0

        MIN=1

        if N>2:

            break

if MAX==1:

    print("The final score of MIN player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

else:

    print("The final score of MAX player is",CurrentScore)

    print("The Final path is",GAME\_PATH)-

#OUTPUT#

Please enter the depth of the tree: 3

15

Enter element 0 (root node)

A

Enter The Score of Staring element

0

Enter left child of element of A

B

Enter The Score of the: B

2

Enter right child of element A

C

Enter The Score of the: C

0

Enter left child of element of B

D

Enter The Score of the: D

-1

Enter right child of element B

E

Enter The Score of the: E

2

Enter left child of element of C

F

Enter The Score of the: F

-5

Enter right child of element C

G

Enter The Score of the: G

0

Enter left child of element of D

H

Enter The Score of the: H

-1

Enter right child of element D

I

Enter The Score of the: I

4

Enter left child of element of E

J

Enter The Score of the: J

2

Enter right child of element E

K

Enter The Score of the: K

6

Enter left child of element of F

L

Enter The Score of the: L

-3

Enter right child of element F

M

Enter The Score of the: M

-5

Enter left child of element of G

N

Enter The Score of the: N

0

Enter right child of element G

O

Enter The Score of the: O

7

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']

SCORES ARE ['0', '2', '0', '-1', '2', '-5', '0', '-1', '4', '2', '6', '-3', '-5', '0', '7']

STARTING STATE OF THE GAME IS A

MOVE 1

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

B

C

C SELECTED!!

MOVE 2

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

F

G

G SELECTED!!

MOVE 3

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

N

O

N SELECTED!!

The final score of MIN player is 0

The Final path is ['C', 'G', 'N']

#P-2

#WHEN MAX PLAYER STARTS THE GAME

GT=[]

SCORE=[]

d = int(input("Please enter the depth of the tree: "))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

GT.append(a)

print("Enter The Score of Staring element")

b=input()

SCORE.append(b)

i=i+1

while i<N:

  print("Enter left child of element of", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  print("Enter right child of element ", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  j=j+1

print(GT)

print("SCORES ARE",SCORE)

MAX=1

MIN=0

MAX=1

MIN=0

currentmove=GT[0]

CurrentScore=SCORE[0]

GAME\_PATH=[]

print("STARTING STATE OF THE GAME IS ",GT[0])

LASTLEVEL=2

N=0

while N<=LASTLEVEL:

    if MAX==0 and MIN==1:

        N=N+1

        print("MOVE ",N)

        print("IT'S MIN  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]<SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=1

        MIN=0

        if N>2:

            break

    if MAX==1 and MIN==0:

        N=N+1

        print("MOVE ",N)

        print("IT'S MAX  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]>SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=0

        MIN=1

        if N>2:

            break

if MAX==1:

    print("The final score of MIN player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

else:

    print("The final score of MAX player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

#OUTPUT#

Please enter the depth of the tree: 3

15

Enter element 0 (root node)

A

Enter The Score of Staring element

23

Enter left child of element of A

B

Enter The Score of the: B

12

Enter right child of element A

C

Enter The Score of the: C

23

Enter left child of element of B

D

Enter The Score of the: D

22

Enter right child of element B

E

Enter The Score of the: E

12

Enter left child of element of C

F

Enter The Score of the: F

25

Enter right child of element C

G

Enter The Score of the: G

23

Enter left child of element of D

H

Enter The Score of the: H

19

Enter right child of element D

I

Enter The Score of the: I

22

Enter left child of element of E

J

Enter The Score of the: J

12

Enter right child of element E

K

Enter The Score of the: K

8

Enter left child of element of F

L

Enter The Score of the: L

25

Enter right child of element F

M

Enter The Score of the: M

8

Enter left child of element of G

N

Enter The Score of the: N

13

Enter right child of element G

O

Enter The Score of the: O

23

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']

SCORES ARE ['23', '12', '23', '22', '12', '25', '23', '19', '22', '12', '8', '25', '8', '13', '23']

STARTING STATE OF THE GAME IS A

MOVE 1

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

B

C

C SELECTED!!

MOVE 2

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

F

G

G SELECTED!!

MOVE 3

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

N

O

O SELECTED!!

The final score of MAX player is 23

The Final path is ['C', 'G', 'O']

#P-3

#WHEN MIN PLAYER STARTS THE GAME

GT=[]

SCORE=[]

d = int(input("Please enter the depth of the tree: "))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

GT.append(a)

print("Enter The Score of Staring element")

b=input()

SCORE.append(b)

i=i+1

while i<N:

  print("Enter left child of element of", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  print("Enter right child of element ", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  j=j+1

print(GT)

print("SCORES ARE",SCORE)

MAX=0

MIN=1

currentmove=GT[0]

CurrentScore=SCORE[0]

GAME\_PATH=[]

print("STARTING STATE OF THE GAME IS ",GT[0])

LASTLEVEL=2

N=0

while N<=LASTLEVEL:

    if MAX==0 and MIN==1:

        N=N+1

        print("MOVE ",N)

        print("IT'S MIN  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]<SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=1

        MIN=0

        if N>2:

            break

    if MAX==1 and MIN==0:

        N=N+1

        print("MOVE ",N)

        print("IT'S MAX  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if SCORE[move1\_pos]>SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

        else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

        GAME\_PATH.append(currentmove)

        MAX=0

        MIN=1

        if N>2:

            break

if MAX==1:

    print("The final score of MIN player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

else:

    print("The final score of MAX player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

#OUTPUT#  
Please enter the depth of the tree: 3

15

Enter element 0 (root node)

A

Enter The Score of Staring element

13

Enter left child of element of A

B

Enter The Score of the: B

19

Enter right child of element A

C

Enter The Score of the: C

13

Enter left child of element of B

D

Enter The Score of the: D

19

Enter right child of element B

E

Enter The Score of the: E

8

Enter left child of element of C

F

Enter The Score of the: F

-8

Enter right child of element C

G

Enter The Score of the: G

13

Enter left child of element of D

H

Enter The Score of the: H

19

Enter right child of element D

I

Enter The Score of the: I

22

Enter left child of element of E

J

Enter The Score of the: J

12

Enter right child of element E

K

Enter The Score of the: K

8

Enter left child of element of F

L

Enter The Score of the: L

25

Enter right child of element F

M

Enter The Score of the: M

-8

Enter left child of element of G

N

Enter The Score of the: N

13

Enter right child of element G

O

Enter The Score of the: O

23

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']

SCORES ARE ['13', '19', '13', '19', '8', '-8', '13', '19', '22', '12', '8', '25', '-8', '13', '23']

STARTING STATE OF THE GAME IS A

MOVE 1

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

B

C

C SELECTED!!

MOVE 2

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

F

G

G SELECTED!!

MOVE 3

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

N

O

N SELECTED!!

The final score of MIN player is 13

The Final path is ['C', 'G', 'N']

#P-3

#WHEN MAX PLAYER STARTS THE GAME

GT=[]

SCORE=[]

d = int(input("Please enter the depth of the tree: "))

N=0

for i in range(0,d+1,1):

  N=N+(2\*\*i)

print(N)

i=0

j=0

print("Enter element ", i, "(root node)")

a=input()

GT.append(a)

print("Enter The Score of Staring element")

b=input()

SCORE.append(b)

i=i+1

while i<N:

  print("Enter left child of element of", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  print("Enter right child of element ", GT[j])

  a=input()

  GT.append(a)

  print("Enter The Score of the: ",a)

  b=input()

  SCORE.append(b)

  i=i+1

  j=j+1

print(GT)

print("SCORES ARE",SCORE)

MAX=1

MIN=0

MAX=1

MIN=0

currentmove=GT[0]

CurrentScore=SCORE[0]

GAME\_PATH=[]

print("STARTING STATE OF THE GAME IS ",GT[0])

GAME\_PATH.append(GT[0])

LASTLEVEL=1

N=0

level=0

loc=0

while N<=LASTLEVEL:

    if MAX==0 and MIN==1:

        N=N+1

        print("MOVE ",N)

        print("IT'S MIN  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if move1\_pos!=loc+(2\*\*level) and move2\_pos!=loc+(2\*\*level)+1:

          print("Error in Values!! Enter them again")

          print("What are the possible moves?")

          move1=input()

          move2=input()

          move1\_pos=GT.index(move1)

          move2\_pos=GT.index(move2)

        else:

          if SCORE[move1\_pos]<SCORE[move2\_pos]:

           print(move1,"SELECTED!!")

           currentmove=move1

           CurrentScore=SCORE[move1\_pos]

          else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

          GAME\_PATH.append(currentmove)

          MAX=1

          MIN=0

          if N>1:

              break

    if MAX==1 and MIN==0:

        N=N+1

        print("MOVE ",N)

        print("IT'S MAX  PLAYER TURN!!!")

        print("What are the possible moves!!")

        move1=input()

        move2=input()

        move1\_pos=GT.index(move1)

        move2\_pos=GT.index(move2)

        if move1\_pos!=loc+(2\*\*level) and move2\_pos!=loc+(2\*\*level) +1:

          print("Error in Values!! Enter them again")

          print("What are the possible moves?")

          move1=input()

          move2=input()

          move1\_pos=GT.index(move1)

          move2\_pos=GT.index(move2)

        else:

          if SCORE[move1\_pos]>SCORE[move2\_pos]:

            print(move1,"SELECTED!!")

            currentmove=move1

            CurrentScore=SCORE[move1\_pos]

          else:

            print(move2,"SELECTED!!")

            currentmove=move2

            CurrentScore=SCORE[move2\_pos]

          GAME\_PATH.append(currentmove)

          MAX=0

          MIN=1

          loc=0

          level=level+1

          for i in range(0,level):

            loc+=(2\*\*i)

          if N>1:

              break

if MAX==1:

    print("The final score of MIN player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

else:

    print("The final score of MAX player is",CurrentScore)

    print("The Final path is",GAME\_PATH)

#OUTPUT#

Please enter the depth of the tree: 2

7

Enter element 0 (root node)

A

Enter The Score of Staring element

3

Enter left child of element of A

B

Enter The Score of the: B

3

Enter right child of element A

C

Enter The Score of the: C

1

Enter left child of element of B

D

Enter The Score of the: D

3

Enter right child of element B

E

Enter The Score of the: E

9

Enter left child of element of C

F

Enter The Score of the: F

1

Enter right child of element C

G

Enter The Score of the: G

7

['A', 'B', 'C', 'D', 'E', 'F', 'G']

SCORES ARE ['3', '3', '1', '3', '9', '1', '7']

STARTING STATE OF THE GAME IS A

MOVE 1

IT'S MAX PLAYER TURN!!!

What are the possible moves!!

B

C

B SELECTED!!

MOVE 2

IT'S MIN PLAYER TURN!!!

What are the possible moves!!

D

E

D SELECTED!!

The final score of MIN player is 3

The Final path is ['A', 'B', 'D']

#P-3

#WHEN MIN PLAYER STARTS THE GAME

Game\_Tree=[]

Score\_Tree=[]

d=int(input("Enter the depth of tree: "))

d2=d

d3=d

n=0

for i in range(0,d+1):

  n=n+(2\*\*i)

a=input("Enter the Root Node: ")

Game\_Tree.append(a)

s=input("Enter the Score of root node: ")

Score\_Tree.append(s)

j=0

i=1

while(i<n):

  print("Enter the left child of the : ",Game\_Tree[j])

  a=input()

  Game\_Tree.append(a)

  print("Enter the score")

  s1=input()

  Score\_Tree.append(s1)

  i+=1

  print("Enter the right child of the : ",Game\_Tree[j])

  b=input()

  Game\_Tree.append(b)

  print("Enter the score")

  s2=input()

  Score\_Tree.append(s2)

  i+=1

  j+=1

print(Game\_Tree)

print(Score\_Tree)

MAX=0

MIN=1

current\_move=Game\_Tree[0]

current\_score=Score\_Tree[0]

Game\_Path=[current\_move]

print("Starting state of the game path is ",Game\_Path)

N=0

level=0

loc=0

while N<=(d3-1):

  if MAX==0 and MIN==1:

    N=N+1

    print("Move ",N)

    print("It's MIN player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    if move1pos!=loc+(2\*\*level) and move2pos!=loc+(2\*\*level)+1:

      print("ERORR IN VALUES TYPE AGAIN")

      print("What are the possible moves?")

      move1=input()

      move2=input()

      move1pos=Game\_Tree.index(move1)

      move2pos=Game\_Tree.index(move2)

    else:

      if Score\_Tree[move1pos]<=Score\_Tree[move2pos]:

        print(move1," Selected!")

        current\_move=move1

        current\_score=Score\_Tree[move1pos]

      else:

        print(move2," Selected!")

        current\_move=move2

        current\_score=Score\_Tree[move2pos]

      Game\_Path.append(current\_move)

      MAX=1

      MIN=0

  else:

    N=N+1

    print("Move ",N)

    print("It's MAX player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    if move1pos!=loc+(2\*\*level) and move2pos!=loc+ (2\*\*level)+1:

      print("ERORR IN VALUES TYPE AGAIN")

      print("What are the possible moves?")

      move1=input()

      move2=input()

      move1pos=Game\_Tree.index(move1)

      move2pos=Game\_Tree.index(move2)

    else:

      if Score\_Tree[move1pos]>Score\_Tree[move2pos]:

        print(move1," Selected!")

        current\_move=move1

        current\_score=Score\_Tree[move1pos]

      else:

        print(move2," Selected!")

        current\_move=move2

        current\_score=Score\_Tree[move2pos]

      Game\_Path.append(current\_move)

      MAX=0

      MIN=1

  loc=0

  level=level+1

  for i in range(0,level):

    loc+=(2\*\*i)

if MAX==1:

  print("Final Score of MIN player is ",current\_score)

  print("PATH IS: ",Game\_Path)

else:

  print("Final Score of MAX player is ",current\_score)

  print("PATH IS: ",Game\_Path)

#OUTPUT#  
Enter the depth of tree: 2

Enter the Root Node: A

Enter the Score of root node: 5

Enter the left child of the : A

B

Enter the score

5

Enter the right child of the : A

C

Enter the score

5

Enter the left child of the : B

D

Enter the score

3

Enter the right child of the : B

E

Enter the score

5

Enter the left child of the : C

F

Enter the score

0

Enter the right child of the : C

G

Enter the score

5

['A', 'B', 'C', 'D', 'E', 'F', 'G']

['5', '5', '5', '3', '5', '0', '5']

Starting state of the game path is ['A']

Move 1

It's MIN player Turn

What are the possible moves?

B

C

B Selected!

Move 2

It's MAX player Turn

What are the possible moves?

D

E

E Selected!

Final Score of MAX player is 5

PATH IS: ['A', 'B', 'E']

#P-4

#WHEN MAX PLAYER STARTS THE GAME

Game\_Tree=[]

Score\_Tree=[]

d=int(input("Enter the depth of tree: "))

d2=d

d3=d

n=0

for i in range(0,d+1):

  n=n+(3\*\*i)

a=input("Enter the Root Node: ")

Game\_Tree.append(a)

s=input("Enter the Score of root node: ")

Score\_Tree.append(s)

j=0

i=1

while(i<n):

  print("Enter the left child of the : ",Game\_Tree[j])

  a=input()

  Game\_Tree.append(a)

  print("Enter the score")

  s1=input()

  Score\_Tree.append(s1)

  i+=1

  print("Enter the middle child of the : ",Game\_Tree[j])

  c=input()

  Game\_Tree.append(c)

  print("Enter the score")

  s3=input()

  Score\_Tree.append(s3)

  i+=1

  print("Enter the right child of the : ",Game\_Tree[j])

  b=input()

  Game\_Tree.append(b)

  print("Enter the score")

  s2=input()

  Score\_Tree.append(s2)

  i+=1

  j+=1

print(Game\_Tree)

print(Score\_Tree)

MAX=1

MIN=0

current\_move=Game\_Tree[0]

current\_score=Score\_Tree[0]

Game\_Path=[current\_move]

print("Starting state of the game path is ",Game\_Path)

N=0

while N<=(d3-1):

  if MAX==0 and MIN==1:

    N=N+1

    print("Move ",N)

    print("It's MIN player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move3=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    move3pos=Game\_Tree.index(move3)

    if Score\_Tree[move1pos]<Score\_Tree[move3pos] and Score\_Tree[move1pos]<Score\_Tree[move3pos]:

      print(move1," Selected!")

      current\_move=move1

      current\_score=Score\_Tree[move1pos]

    elif Score\_Tree[move2pos]<Score\_Tree[move1pos] and Score\_Tree[move2pos]<Score\_Tree[move3pos]:

      print(move2," Selected!")

      current\_move=move2

      current\_score=Score\_Tree[move2pos]

    else:

      print(move3," Selected!")

      current\_move=move3

      current\_score=Score\_Tree[move3pos]

    Game\_Path.append(current\_move)

    MAX=1

    MIN=0

  else:

    N=N+1

    print("Move ",N)

    print("It's MAX player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move3=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    move3pos=Game\_Tree.index(move3)

    if Score\_Tree[move1pos]>=Score\_Tree[move2pos] and Score\_Tree[move1pos]>=Score\_Tree[move3pos]:

      print(move1," Selected!")

      current\_move=move1

      current\_score=Score\_Tree[move1pos]

    elif Score\_Tree[move2pos]>Score\_Tree[move1pos] and Score\_Tree[move2pos]>Score\_Tree[move3pos]:

      print(move2," Selected!")

      current\_move=move2

      current\_score=Score\_Tree[move2pos]

    else:

      print(move3," Selected!")

      current\_move=move3

      current\_score=Score\_Tree[move3pos]

    Game\_Path.append(current\_move)

    MAX=0

    MIN=1

if MAX==1:

  print("Final Score of MIN player is ",current\_score)

  print("PATH IS: ",Game\_Path)

else:

  print("Final Score of MAX player is ",current\_score)

  print("PATH IS: ",Game\_Path)

#OUTPUT#

Enter the depth of tree: 2

Enter the Root Node: A

Enter the Score of root node: -3

Enter the left child of the : A

B

Enter the score

-4

Enter the middle child of the : A

C

Enter the score

-3

Enter the right child of the : A

D

Enter the score

-4

Enter the left child of the : B

E

Enter the score

-2

Enter the middle child of the : B

F

Enter the score

-4

Enter the right child of the : B

H

Enter the score

3

Enter the left child of the : C

I

Enter the score

-3

Enter the middle child of the : C

J

Enter the score

-1

Enter the right child of the : C

K

Enter the score

1

Enter the left child of the : D

L

Enter the score

-4

Enter the middle child of the : D

M

Enter the score

1

Enter the right child of the : D

N

Enter the score

-3

['A', 'B', 'C', 'D', 'E', 'F', 'H', 'I', 'J', 'K', 'L', 'M', 'N']

['-3', '-4', '-3', '-4', '-2', '-4', '3', '-3', '-1', '1', '-4', '1', '-3']

Starting state of the game path is ['A']

Move 1

It's MAX player Turn

What are the possible moves?

B

C

D

B Selected!

Move 2

It's MIN player Turn

What are the possible moves?

F

E

H

F Selected!

Final Score of MIN player is -4

PATH IS: ['A', 'B', 'F']

#P-4

#WHEN MIN PLAYER STARTS THE GAME

Game\_Tree=[]

Score\_Tree=[]

d=int(input("Enter the depth of tree: "))

d2=d

d3=d

n=0

for i in range(0,d+1):

  n=n+(3\*\*i)

a=input("Enter the Root Node: ")

Game\_Tree.append(a)

s=input("Enter the Score of root node: ")

Score\_Tree.append(s)

j=0

i=1

while(i<n):

  print("Enter the left child of the : ",Game\_Tree[j])

  a=input()

  Game\_Tree.append(a)

  print("Enter the score")

  s1=input()

  Score\_Tree.append(s1)

  i+=1

  print("Enter the middle child of the : ",Game\_Tree[j])

  c=input()

  Game\_Tree.append(c)

  print("Enter the score")

  s3=input()

  Score\_Tree.append(s3)

  i+=1

  print("Enter the right child of the : ",Game\_Tree[j])

  b=input()

  Game\_Tree.append(b)

  print("Enter the score")

  s2=input()

  Score\_Tree.append(s2)

  i+=1

  j+=1

print(Game\_Tree)

print(Score\_Tree)

MAX=0

MIN=1

current\_move=Game\_Tree[0]

current\_score=Score\_Tree[0]

Game\_Path=[current\_move]

print("Starting state of the game path is ",Game\_Path)

N=0

while N<=(d3-1):

  if MAX==0 and MIN==1:

    N=N+1

    print("Move ",N)

    print("It's MIN player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move3=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    move3pos=Game\_Tree.index(move3)

    if Score\_Tree[move1pos]<Score\_Tree[move3pos] and Score\_Tree[move1pos]<Score\_Tree[move3pos]:

      print(move1," Selected!")

      current\_move=move1

      current\_score=Score\_Tree[move1pos]

    if Score\_Tree[move2pos]<=Score\_Tree[move1pos] and Score\_Tree[move2pos]<=Score\_Tree[move3pos]:

      print(move2," Selected!")

      current\_move=move2

      current\_score=Score\_Tree[move2pos]

    else:

      print(move3," Selected!")

      current\_move=move3

      current\_score=Score\_Tree[move3pos]

    Game\_Path.append(current\_move)

    MAX=1

    MIN=0

  else:

    N=N+1

    print("Move ",N)

    print("It's MAX player Turn")

    print("What are the possible moves?")

    move1=input()

    move2=input()

    move3=input()

    move1pos=Game\_Tree.index(move1)

    move2pos=Game\_Tree.index(move2)

    move3pos=Game\_Tree.index(move3)

    if Score\_Tree[move1pos]>Score\_Tree[move2pos] and Score\_Tree[move1pos]>Score\_Tree[move3pos]:

      print(move1," Selected!")

      current\_move=move1

      current\_score=Score\_Tree[move1pos]

    elif Score\_Tree[move2pos]>Score\_Tree[move1pos] and Score\_Tree[move2pos]>Score\_Tree[move3pos]:

      print(move2," Selected!")

      current\_move=move2

      current\_score=Score\_Tree[move2pos]

    else:

      print(move3," Selected!")

      current\_move=move3

      current\_score=Score\_Tree[move3pos]

    Game\_Path.append(current\_move)

    MAX=0

    MIN=1

if MAX==1:

  print("Final Score of MIN player is ",current\_score)

  print("PATH IS: ",Game\_Path)

else:

  print("Final Score of MAX player is ",current\_score)

  print("PATH IS: ",Game\_Path)

#OUTPUT#

Enter the depth of tree: 2

Enter the Root Node: A

Enter the Score of root node: 1

Enter the left child of the : A

B

Enter the score

3

Enter the middle child of the : A

C

Enter the score

1

Enter the right child of the : A

D

Enter the score

1

Enter the left child of the : B

E

Enter the score

-2

Enter the middle child of the : B

F

Enter the score

-4

Enter the right child of the : B

H

Enter the score

3

Enter the left child of the : C

I

Enter the score

-3

Enter the middle child of the : C

J

Enter the score

-1

Enter the right child of the : C

K

Enter the score

1

Enter the left child of the : D

L

Enter the score

-4

Enter the middle child of the : D

M

Enter the score

1

Enter the right child of the : D

N

Enter the score

-3

['A', 'B', 'C', 'D', 'E', 'F', 'H', 'I', 'J', 'K', 'L', 'M', 'N']

['1', '3', '1', '1', '-2', '-4', '3', '-3', '-1', '1', '-4', '1', '-3']

Starting state of the game path is ['A']

Move 1

It's MIN player Turn

What are the possible moves?

B

C

D

C Selected!

Move 2

It's MAX player Turn

What are the possible moves?

I

J

K

K Selected!

Final Score of MAX player is 1

PATH IS: ['A', 'C', 'K']

**LAB 10**

# !pip install pomegranate

import math

from pomegranate import \*

Rain=DiscreteDistribution({'T':0.4,'F':0.6})

WaterLog=ConditionalProbabilityTable([['T','T',0.8],['T','F',0.2],['F','T',0.3],['F','F',0.7]],[Rain])

Traffic=ConditionalProbabilityTable([['T','T',0.6],['T','F',0.4],['F','T',0.2],['F','F',0.3]],[Rain])

R=Node(Rain,name='Rain')

W=Node(WaterLog,name="WaterLog")

H=Node(Traffic,name="Heavy Traffic")

network=BayesianNetwork("NETWORK!")

network.add\_states(R,W,H)

network.add\_edge(R,W)

network.add\_edge(R,H)

network.bake()

print(network.probability([['T','T','T'],['T','T','F'],['T','F','T']]))

#OUTPUT#  
[0.192 0.128 0.048]

# !pip install pomegranate

import math

from pomegranate import \*

Rain=DiscreteDistribution({'T':0.4,'F':0.6})

WaterLog=ConditionalProbabilityTable([['T','T',0.8],['T','F',0.2],['F','T',0.3],['F','F',0.7]],[Rain])

Traffic=ConditionalProbabilityTable([['T','T',0.6],['T','F',0.4],['F','T',0.2],['F','F',0.3]],[Rain])

College=ConditionalProbabilityTable([['T','T','T',0.8],

                                     ['T','T','F',0.2],

                                     ['T','F','T',0.5],

                                     ['T','F','F',0.5],

                                     ['F','T','T',0.6],

                                     ['F','T','F',0.4],

                                     ['F','F','T',0.3],

                                     ['F','F','F',0.7]],

                                     [WaterLog,Traffic])

R=Node(Rain,name='Rain')

W=Node(WaterLog,name="WaterLog")

H=Node(Traffic,name="Heavy Traffic")

C=Node(College,name="Late to college")

network=BayesianNetwork("NETWORK!")

network.add\_states(R,W,H,C)

network.add\_edge(R,W)

network.add\_edge(R,H)

network.add\_edge(W,C)

network.add\_edge(H,C)

network.bake()

# print(network)

print(network.probability([['T','T','T','T'],['T','T','T','F'],['T','T','F','T']]))

#OUTPUT#

[0.1536 0.0384 0.064 ]

import math

from pomegranate import \*

Burglary=DiscreteDistribution({'T':0.002,'F':0.998}) #B

Fire=DiscreteDistribution({'T':0.001,'F':0.999}) #F

Alarm=ConditionalProbabilityTable([['T','T','T',0.94],

                                     ['T','T','F',0.06],

                                     ['T','F','T',0.95],

                                     ['T','F','F',0.05],

                                     ['F','T','T',0.69],

                                     ['F','T','F',0.31],

                                     ['F','F','T',0.999],

                                     ['F','F','F',0.001]],

                                     [Burglary,Fire])  #A

Neighbour=ConditionalProbabilityTable([['T','T',0.91],['T','F',0.09],['F','T',0.05],['F','F',0.95]],[Alarm]) #N

Message=ConditionalProbabilityTable([['T','T',0.75],['T','F',0.25],['F','T',0.02],['F','F',0.98]],[Alarm]) #M

B=Node(Burglary,name='Chori Ho gyi')

F=Node(Fire,name="Aag lag gyi")

A=Node(Alarm,name="Alarm")

N=Node(Neighbour,name="Pasdosi")

M=Node(Message,name="Message")

network=BayesianNetwork("NETWORK!")

network.add\_states(B,F,A,N,M)

network.add\_edge(B,A)

network.add\_edge(F,A)

network.add\_edge(A,N)

network.add\_edge(A,M)

network.bake()

print(network.probability([['T','T','T','T','T'],['T','T','T','T','F'],['T','T','T','F','T']]))

#OUTPUT#

[1.2831e-06 4.2770e-07 1.2690e-07]

import math

from pomegranate import \*

Blood\_Pressure=DiscreteDistribution({'T':0.2,'F':0.8}) #B

Diabetes=DiscreteDistribution({'T':0.5,'F':0.5}) #D

Heart\_Disease=ConditionalProbabilityTable([

                                            ['T','T','T',0.8],

                                            ['T','T','F',0.2],

                                            ['T','F','T',0.6],

                                            ['T','F','F',0.5],

                                            ['F','T','T',0.4],

                                            ['F','T','F',0.6],

                                            ['F','F','T',0.1],

                                            ['F','F','F',0.9]],

                                            [Blood\_Pressure,Diabetes])  #H

Cardiac\_Arrest=ConditionalProbabilityTable([['T','T',0.91],['T','F',0.09],['F','T',0.05],['F','F',0.95]],[Heart\_Disease]) #C

B=Node(Blood\_Pressure,name='Blood\_Pressure')

D=Node(Diabetes,name="Diabetes")

C=Node(Cardiac\_Arrest,name="Cardiac\_Arrest")

H=Node(Heart\_Disease,name="Heart\_Disease")

network=BayesianNetwork("NETWORK!")

network.add\_states(B,D,H,C)

network.add\_edge(B,H)

network.add\_edge(D,H)

network.add\_edge(H,C)

network.bake()

print(network.probability([['T','T','T','T'],['T','T','T','F'],['T','T','F','T']]))

#OUTPUT#  
[0.0728 0.0072 0.001 ]

**LAB 11**

##PROGRAM 1

class Sum:

  def \_\_init\_\_(self,lst):

    self.sum=0;

    self.do\_sum(lst);

  def do\_sum(self,lst):

    for x in lst:

      self.sum+=x

  def get\_sum(self):

    print( self.sum)

s=Sum([a for a in range(1,11)])

s.get\_sum()

#OUTPUT#

55

## PROGRAM 2

class Factorial:

 def \_\_init\_\_(self,num):

   self.value=1

   for a in range(1,num+1):

     self.value\*=a

var=Factorial(10)

print(var.value)

#OUTPUT

3628800

##PROGRAM 3

class Calculator:

  def \_\_init\_\_(self):

    self.value=0

  def add(self,num1,num2):

    print(num1,"+",num2,"= ",num1+num2)

  def sub(self,num1,num2):

    print(num1,"-",num2,"= ",num1-num2)

  def div(self,num1,num2):

    print(num1,"/",num2,"= ",num1/num2)

  def mult(self,num1):

    print(num1,"\*",num2,"= ",num1\*num2)

  def menu(self):

    num1=float(input("Enter Number 1 : "))

    num2=float(input("Enter Number 2 : "))

    print("1:Add  2:Subtraction   3:Divide    4:Multiply")

    ch=int(input("Enter Choice : "))

    if (ch==1):

      self.add(num1,num2)

    elif (ch==2):

      self.sub(num1,num2)

    elif (ch==3):

      self.div(num1,num2)

    elif (ch==4):

      self.mult(num1,num2)

    else:

      print("Wrong input")

c=Calculator()

c.menu()

#OUTPUT#

Enter Number 1 : 22

Enter Number 2 : 7

1:Add 2:Subtraction 3:Divide 4:Multiply

Enter Choice : 3

22.0 / 7.0 = 3.142857142857143

##PROGRAM 4

class CLASS1:

  def \_\_init\_\_(self,num1=0,num2=0):

    self.num1=num1

    self.num2=num2

class CLASS2(CLASS1):

  def \_\_init\_\_(self,num1=0,num2=0):

      super().\_\_init\_\_(num1,num2)

  def add(self):

    return self.num1+self.num2

class CLASS3(CLASS2):

  def \_\_init\_\_(self,num1=0,num2=0):

    super().\_\_init\_\_(num1,num2)

  def sub(self):

    return self.num1-self.num2

obj1= CLASS1(10,20)

obj2= CLASS2(100,200)

obj3= CLASS3(1000,2000)

print("Num 1 : ",obj1.num1,"\nNum 2 : ",obj1.num2)

print(obj2.add())

print(obj3.sub())

#OUTPUT#

Num 1 : 10

Num 2 : 20

300

-1000

Finally Finished :)

Made By- Avishkaar Pawar

Roll No- AD-1224