**ARTIFICIAL INTELLIGENCE LAB RECORD**

**WEEK-1**

**1 Generate a Calendar for the given month and year?**

**CODE:**

import calendar

x=int(input("enter the month: "))

y=int(input("enter the year: "))

z=calendar.month(y,x)

print(z)

**OUTPUT**

**A screenshot of a computer

Description automatically generated with low confidence**

**1b Implement a Simple Calculator program?**

**CODE:**

def add(a,b):

  return a+b

def sub(a,b):

  return a-b

def multiply(a,b):

  return a\*b

def division(a,b):

  return a/b

print("1. Addition\n2. Subtraction\n3. Multiplication\n4. Division")

a=int(input("enter first number: "))

b=int(input("enter second number: "))

d=int(input("enter the choice: "))

if d==1:

  print("the addition of two numbers is: ",add(a,b))

elif d==2:

  print("the subtraction of two numbers is: ",sub(a,b))

elif d==3:

  print("the product of two numbers is: ",multiply(a,b))

elif d==4:

  print("the division of two numbers is: ",division(a,b))

else:

  print("Exit")

**OUTPUT:**

**Text

Description automatically generated Graphical user interface, text, application

Description automatically generated**

**A screenshot of a computer

Description automatically generated with medium confidence** **Text

Description automatically generated**

**WEEK-2**

**2. Design of intelligent systems.(suggested excersice to control the VACUUM cleaner moves)**

**Code:**

clean = 0

dirt = 1

a = 0

b = 0

def testing(loc,s,cost):

   if(s==dirt):

     print("Room "+loc+" is dirty")

     cost += 1

     s = clean

     print("Room "+loc+" is Cleaned by vaccum cleaner")

   else :

     print("Room "+loc+" is already Clean")

   return s,cost

global cost

cost = 0

loc = input("Enter Location : ")

a = int(input("enter status of Room A : "))

b = int(input("enter status of Room B : "))

if loc is 'A':

  print("Current room A")

  a,cost = testing(loc,a,cost)

  loc = 'B'

  if b==dirt:

    print("Moving to room B")

    cost += 1

    b,cost = testing(loc,b,cost)

  else:

    print("Room "+loc+" is clean")

else :

  b,cost = testing(loc,b,cost)

  print("Current room B")

  loc = 'A'

  if a==1:

    print("Moving to room A")

    cost += 1

    a,cost = testing(loc,a,cost)

  else:

    print("Room "+loc+" is clean")

if a==0 and b==0 :

  print("Room A and B are clean and cost of cleaning is ",cost)

**Output:**

**Text

Description automatically generated**

**WEEK-3**

3) Implement the production system and derive a solution for the real world AI problem. (Suggested exercise: Write a program to solve the following problem: You have two jugs, a 4- gallon and a 3-gallon. Neither of the jugs has markings on them. There is a pump that can be used to fill the jugs with water. How can you get exactly two gallons of water in the 4-gallon jug?).

CODE:

import math

def check():

 if need>max(j1,j2):

   return "No"

 if need % math.gcd(j1,j2)==0:

   return "Yes"

 return "No"

j1,j2=map(int,input("Enter the capacity of Jug 1 and Jug 2: ").split())

need=int(input("Enter the capacity of water needed: "))

x, y=0, 0

if check()=="No":

 print("Problem can't be solved")

else:

 while x!=need and y!=need:

   if x==0:

     x=j1

   elif x>0 and y!=j2:

     while x!=0 and y!=j2:

       x,y=x-1,y+1

   elif x>0 and y==j2:

     y=0

   print("JUG1:",x,"JUG2:",y)

 if x==need:

   print("JUG1 Contains the needed quantity")

 else:

   print("JUG2 Contains the needed quantity")

**Output:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**WEEK-4**

4) Implement A\* algorithm. (Suggested exercise: to find the shortest path)

**Code:**

n=int(input("Enter no. of nodes: "))

heuristic=[0]+list(map(int,input("Enter Heuristic values: ").split()))

cost=[0,]

path=[]

for x in range(n):

 cost.append([0]+list(map(int,input().split())))

sour=int(input("Enter the source: "))

dest=int(input("Enter the destination: "))

current=sour

sum=0

while current!=dest:

 min=9999

 minval=0

 for i in range(1,n+1):

   x=heuristic[i]+cost[current][i]

   if x<min and i not in path:

     min=x

     minval=i

 sum+=min

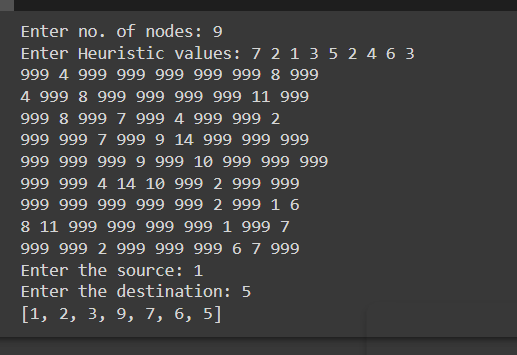
 path.append(current)

 current=minval

path.append(dest)

print(path)

**Output:**

****

**WEEK-5**

5) Implement the Constraint Specific Problem. (Suggested exercise: a crossword puzzle).

**Code:**

import math

from itertools import permutations

def fun(a):

 sum1=0

 p=0

 for x in a[::-1]:

   sum1+=d[x]\*int(math.pow(10,p))

   p+=1

 return sum1

a=input("Enter the 1st String: ")

b=input("Enter the 2nd String: ")

r=input("Enter result: ")

d={}

for x in a+b+r:

 if x not in d.keys():

   d[x]=0

perm = permutations([0,1,2,3,4,5,6,7,8,9],len(d))

for i in list(perm):

 index=0

 for x in d.keys():

   d[x]=i[index]

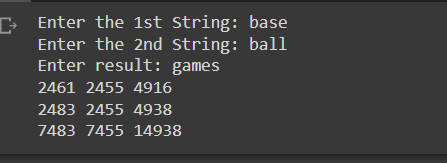
   index+=1

 x,y,z=fun(a),fun(b),fun(r)

 if x+y==z:

   print(x,y,z)

**Output:**

****

**WEEK-6**

6) Implement the alpha-beta pruning. (Suggested exercise: for a tic toc game).

**Code:**

matrix = [i for i in range(9)]

def availability(c):

    if matrix[c] == c :

        return True

    else :

        return False

def checkboard():

    for i in range(0,9,3):

        if(matrix[i]==matrix[i+1] and matrix[i+1]==matrix[i+2]):

            return 1

    for i in range(3):

        if(matrix[i]==matrix[i+3] and matrix[i+3]==matrix[i+6]):

            return 1

    if(matrix[0] == matrix[4] and matrix[4]==matrix[8]):

        return 1

    if(matrix[2]==matrix[4] and matrix[4]==matrix[6]):

        return 1

    for i in range(9):

        if matrix[i] == i:

            return 0

    return -1

def display():

    print("-------------")

    print("| ",end="")

    for i in range(9):

        if i%3 == 0 and i!=0:

            print()

            print("-------------")

            print("| ",end="")

        print(matrix[i],end=" | ")

    print()

    print("-------------")

c = 0

s1 = input("enter symbol of player 1 : ")

s2 = input("enter symbol of player 2 : ")

turn = 0

display()

while(c!=9):

    choice = int(input("enter player "+str(turn+1)+" choice position : "))

    if(availability(choice)):

        matrix[choice] = s1 if turn ==0 else s2

        c += 1

        display()

        result = checkboard()

        if(result == 1):

            print("player "+str(turn+1)+" wins ")

            break

        elif result == -1:

            print("draw")

            break

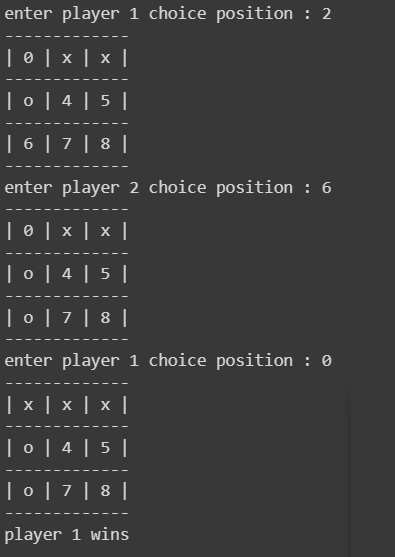
        turn = (turn+1)%2

    else :

        print("Invalid Choice")

**Output:**

**Text

Description automatically generated with medium confidence **