**Text, logo, company name

Description automatically generated LAB RECORD**

**BACHELOR OF TECHNOLOGY**

**B.Tech (CSE) Semester (VI)**

**(Academic Session – 2021 - 2022)**

**Course Title : Artificial Intelligence**

**Course Code : CSE401**

**Enrollment No. : A7605219040**

**Name of Student : RAUNAK SIKKA**

**Date of Submission : \_\_ \_\_ / \_\_ \_\_ /\_\_ \_\_ \_\_ \_\_**

**Signature of Student :**

**Grade/Marks Obtained :**

**Faculty Name & Signature : Dr. Pooja Khanna**

**Department of Computer Science & Engineering**

**Amity School of Engineering & Technology**

**Amity University, Lucknow Campus**

**Department of Computer Science & Engineering**

**Amity School of Engineering & Technology**

**Amity University, Lucknow Campus**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **Program Title** | **Date** | **Signature** |
| **1** | **Some basic python problems** |  |  |
| **2** | **Write a python program to implement stack.** |  |  |
| **3** | **Write a python program for tic tac toe problem.** |  |  |
| **4** | **Write a python program to implement BFS for water jug problem.** |  |  |
| **5** | **Write a python program for binary knapsack.** |  |  |
| **6** | **Write a python program for evaluation for Boolean expression.** |  |  |

**Program-1**

1. Write a Python program which accepts the user's first and last name and print them in reverse order with a space between them

Graphical user interface, text

Description automatically generated

OUTPUT:

Graphical user interface, text

Description automatically generated

2. Write a Python program to find whether a given number (accept from the user) is even or odd, print out an appropriate message to the user.

Text

Description automatically generated

OUTPUT:

Graphical user interface, text

Description automatically generated

3. Write a Python program which accepts the radius of a circle from the user and compute the area.

Text

Description automatically generated

OUTPUT:

Text

Description automatically generated

4. Write a Python program to get the difference between a given number and 17, if the number is greater than 17 return double the absolute difference.

Text

Description automatically generated

OUTPUT:

Text

Description automatically generated with medium confidence

5. Write a Python program to calculate the sum of three given numbers, if the values are equal then return thrice of their sum

Text

Description automatically generated

OUTPUT:

Graphical user interface, text

Description automatically generated

6. Write a Python program to count the number 4 in a given list.

Text

Description automatically generated

OUTPUT:

Text

Description automatically generated with medium confidence

7. Write a Python program to compute the distance between the points (x1, y1) and (x2, y2).

Text

Description automatically generated

OUTPUT:

Graphical user interface, text

Description automatically generated

8. Write a Python program to print out all even numbers from a given numbers list in the same order and stop the printing if any numbers that come after 237 in the sequence.

numbers = [

386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328, 615, 953, 345,

399, 162, 758, 219, 918, 237, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217,

815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445, 742, 717,

958,743, 527 ]

Text

Description automatically generated

OUTPUT:

Text

Description automatically generated

**PROGRAM-2**

**Write a python program to implement stack.**

def isEmpty(stk):

    if stk==[]:

        return True

    else:

        return False

def Push(stk, item):

    stk.append(item)

    top=len(stk)-1

def Pop(stk):

    if isEmpty(stk):

        return "Underflow"

    else:

        item=stk.pop()

        if len(stk)==0:

            top=None

        else:

            top=len(stk)-1

        return item

def Peek(stk):

    if isEmpty(stk):

        return "Underflow"

    else:

        top=len(stk)-1

    return item

def Peek(stk):

    if isEmpty(stk):

        return "Underflow"

    else:

        top=len(stk)-1

        return stk[top]

def disp(stk):

    if isEmpty(stk):

        print("Stack Empty")

    else:

        top=len(stk)-1

        print(stk[top],"<-top")

        for a in range(top-1,-1,-1):

            print(stk[a])

stack=[]

top=None

while True:

    print("\n\nStack Operations:")

    print("1. Push")

    print("2. Pop")

    print("3. Peek")

    print("4. Display Stack")

    print("5. Exit")

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

    if ch==1:

        item=int(input("Enter Item:"))

        Push(stack,item)

    elif ch==2:

        item=Pop(stack)

        if item=="Underflow":

            print("stack is empty")

        else:

            print("Popped item is : ",item)

    elif ch==3:

        item=Peek(stack)

        if item=="Underflow":

            print("Stack is empty")

        else:

            print("Topmost item is : ",item)

    elif ch==4:

        disp(stack)

    elif ch==5:

        break

    else:

        print("Invalid Choice ")

**Output:**

Text

Description automatically generated

**PROGRAM-3**

**Write a python program for tic tac toe problem.**

board={"T1":" ","T2":" ","T3":" ","M1":" ","M2":" ","M3":" ","D1":" ","D2":" ","D3":" "}

player=1

total\_moves=0

end\_check=0

print("T1|T2|T3")

print("- +- +- ")

print("M1|M2|M3")

print("- +- +- ")

print("D1|D2|D3")

print("- +- +- ")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

def check():

if board["T1"]=="X" and board["T2"]=="X" and board["T3"]=="X":

print("player-1 winned the game")

return 1

if board["M1"]=="X" and board["M2"]=="X" and board["M3"]=="X":

print("player-1 winned the game")

return 1

if board["D1"]=="X" and board["D2"]=="X" and board["D3"]=="X":

print("player-1 winned the game")

return 1

if board["T1"]=="X" and board["M1"]=="X" and board["D1"]=="X":

print("player-1 winned the game")

return 1

if board["M2"]=="X" and board["T2"]=="X" and board["D2"]=="X":

print("player-1 winned the game")

return 1

if board["T3"]=="X" and board["M3"]=="X" and board["D3"]=="X":

print("player-1 winned the game")

return 1

if board["T1"]=="X" and board["M2"]=="X" and board["D3"]=="X":

print("player-1 winned the game")

return 1

if board["T3"]=="X" and board["M2"]=="X" and board["D1"]=="X":

print("player-1 winned the game")

return 1

if board["T1"]=="O" and board["T2"]=="O" and board["T3"]=="O":

print("player-2 winned the game")

return 1

if board["M1"]=="O" and board["M2"]=="O" and board["M3"]=="O":

print("player-2 winned the game")

return 1

if board["D1"]=="O" and board["D2"]=="O" and board["D3"]=="O":

print("player-2 winned the game")

return 1

if board["T1"]=="O" and board["M1"]=="O" and board["D1"]=="O":

print("player-2 winned the game")

return 1

if board["M2"]=="O" and board["T2"]=="O" and board["D2"]=="O":

print("player-2 winned the game")

return 1

if board["T3"]=="O" and board["M3"]=="O" and board["D3"]=="O":

print("player-2 winned the game")

return 1

if board["T1"]=="O" and board["M2"]=="O" and board["D3"]=="O":

print("player-2 winned the game")

return 1

if board["T3"]=="O" and board["M2"]=="O" and board["D1"]=="O":

print("player-2 winned the game")

return 1

while True:

print(board["T1"]+'|'+board["T2"]+'|'+board["T3"])

print("-+-+-")

print(board["M1"]+'|'+board["M2"]+'|'+board["M3"])

print("-+-+-")

print(board["D1"]+'|'+board["D2"]+'|'+board["D3"])

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

end\_check=check()

if total\_moves==9:

print("game over")

if total\_moves==9 or end\_check==1:

break

while True:

if player==1:

p1\_input=input("player one-give the position")

if p1\_input.upper() in board and board[p1\_input.upper()]==" ":

board[p1\_input.upper()]="X"

total\_moves = total\_moves + 1

player=2

break

else:

print("invalid position, please try again")

continue

else:

p2\_input=input("player two- give the position")

if p2\_input.upper() in board and board[p2\_input.upper()]==" ":

board[p2\_input.upper()]="O"

total\_moves = total\_moves + 1

player=1

break

else:

print("invalid position, please try again")

continue

**Output:**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

**PROGRAM-4**

**Write a python program to implement BFS for water jug problem.**

jug1, jug2, aim = 4, 3, 2

# Initialize dictionary with

# default value as false.

visited = []

def waterJugSolver(amt1, amt2):

# Checks for our goal and

# returns true if achieved.

if (amt1 == aim and amt2 == 0) or (amt2 == aim and amt1 == 0):

print(amt1, amt2)

# Checks if we have already visited the

# combination or not. If not, then it proceeds further.

if (amt1,amt2) not in visited:

print(amt1, amt2)

# Changes the boolean value of

# the combination as it is visited.

visited.append((amt1,amt2))

# Check for all the 6 possibilities and

# see if a solution is found in any one of them.

return (waterJugSolver(0, amt2) or

waterJugSolver(amt1, 0) or

waterJugSolver(jug1, amt2) or

waterJugSolver(amt1, jug2) or

waterJugSolver(amt1 + min(amt2, (jug1-amt1)),

amt2 - min(amt2, (jug1-amt1))) or

waterJugSolver(amt1 - min(amt1, (jug2-amt2)),

amt2 + min(amt1, (jug2-amt2))))

# Return False if the combination is

# already visited to avoid repetition otherwise

# recursion will enter an infinite loop.

else:

return False

print("Steps: ")

# Call the function and pass the

# Initial amount of water present in both jugs.

waterJugSolver(0, 0)

**Output:**

Text

Description automatically generated

**PROGRAM-5**

**Write a python program for binary knapsack.**

def knapSack(W, wt, val, n):

if n == 0 or W == 0 :

return 0

if (wt[n-1] > W):

return knapSack(W, wt, val, n-1)

else:

return max(val[n-1] + knapSack(W-wt[n-1], wt, val, n-1),

knapSack(W, wt, val, n-1))

val = [50,100,150,200]

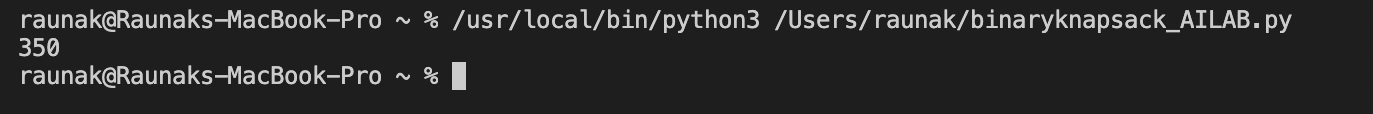
wt = [8,16,32,40]

W = 64

n = len(val)

print (knapSack(W, wt, val, n))

**Output:**



**PROGRAM-6**

**Write a python program for evaluation for Boolean expression.**

def truthTable(expression,inputs=2):

  print("Boolean Expression:")

  print("  X = " + expression.upper())

  expression = expression.lower()

  #replace Boolean Operators with bitwise operators

  expression = expression.replace("and","&")

  expression = expression.replace("xor","^")

  expression = expression.replace("or","|")

  expression = expression.replace("not","~")

  print("\nTruth Table:")

  if inputs==2:

    print("  -------------")

    print("  | A | B | X |")

    print("  -------------")

    for a in range(0,2):

      for b in range(0,2):

        x = eval(expression)

        print("  | " + str(a) + " | " + str(b) + " | " + str(x) + " |" )

        print("  -------------")

  elif inputs==3:

    print("  -----------------")

    print("  | A | B | C | X |")

    print("  -----------------")

    for a in range(0,2):

      for b in range(0,2):

        for c in range(0,2):

          x = eval(expression)

          print("  | " + str(a) + " | " + str(b) + " | " + str(c) + " | " + str(x) + " |" )

          print("  -----------------")

  elif inputs==4:

    print("  ---------------------")

    print("  | A | B | C | D | X |")

    print("  ---------------------")

    for a in range(0,2):

      for b in range(0,2):

        for c in range(0,2):

          for d in range(0,2):

            x = eval(expression)

            print("  | " + str(a) + " | " + str(b) + " | " + str(c) + " | " + str(d) + " | " + str(x) + " |" )

            print("  ---------------------")

##############################################

expression = "A AND NOT (B XOR C)"

truthTable(expression,3)

**Output:**

**Text

Description automatically generated**