Logo, company name

Description automatically generated

**DATA STRUCTURE AND ALGORITHMS**

**Date : 23 October, 2021**

**Lab no : 3**

**Name : Shumail Inam**

**Reg no : 200901032**

**Lab task # 1**

Write a program to let the user enter a string of his own choice. Check whether the given string is a palindrome or not.

**CODE using stack:**

class Stack:                    # making a stack class

    def \_\_init\_\_(self):             #defining (item) function of stack

        self.items = []

    def is\_empty(self):              #defining (is\_empty) function of stack

        return self.items == []

    def push(self, data):            #defining (push) function of a stack

        self.items.append(data)

    def pop(self):                #defining (pop) function of a stack

        return self.items.pop()

  a=input("enter a string")           #taking input from user

b=Stack()             #making a stack

rev\_a=''

for ch in a:

   b.push(ch)           #pushing chracters in stack

while not b.is\_empty():

  rev\_a=rev\_a+ b.pop()    #poping characters from stack and storing it in a variable

if rev\_a==a:               # comparing string with variable

  print("given string is a palindrome")

else:

  print("given string is not a palindrome")

**OUTPUT**

enter a string mam

given string is a palindrome

**CODE using deque:**

from collections import deque         #importing library (deque)

a=input("enter a string:")

stack=deque()                         #creating a list

rev\_a=''

for ch in a:

  stack.append(ch)                  # pushing chracters in list

while (len(stack)!=0):

  rev\_a=rev\_a+stack.pop()

if a==rev\_a:

  print("given string is a palindrome")

else:

  print("given string is not a palindrome")

**OUTPUT**

enter a string:abcd

given string is not a palindrome

**Lab task # 2**

Write a program to check the balanced parenthesis in the expression or not using stack.

**CODE**

from collections import deque

a = input("Enter any equation: ")          # taking input from user

stack = deque()

count = 0

for ch in a:                       # using loop to get chracters from equation

    if (ch == '(' or ch == '[' or ch == '{'):

        stack.append(ch)         #pushing chracters in stack

        count = count + 1

                            # incrementing count if first parenthesis is found

    elif (ch == ')' or ch == ']' or ch == '}'):

        if (len(stack) != 0):

            if (ch == ')'):

                if(stack[-1] == '('):

                    count = count - 1

                             #decrementing count if opposite parenthesis is found

            if (ch == ']'):

                if (stack[-1] == '['):

                    count = count - 1

            if (ch == '}'):

                if (stack[-1] == '{'):

                    count = count - 1

            stack.pop()           #poping chracters from stack

        if(len(a)==0):

          print("invalid syntax")

print(count)

if (count > 0):

   print("unbalanced equation")

else:

    print("balanced syntax")

**OUTPUT**

enter a equation:{()}

0

balanced syntax