**Lab 10**

**Name: Omerullah Ansari**

**ID: 65584**

**Q1**

**def prec(c):**

**if c == '^':**

**return 3**

**elif c == '/' or c == '\*':**

**return 2**

**elif c == '+' or c == '-':**

**return 1**

**else:**

**return -1**

**def associativity(c):**

**if c == '^':**

**return 'R'**

**return 'L'**

**def infixToPostfix(s):**

**result = []**

**stack = []**

**for i in range(len(s)):**

**c = s[i]**

**if ('a' <= c <= 'z') or ('A' <= c <= 'Z') or ('0' <= c <= '9'):**

**result.append(c)**

**elif c == '(':**

**stack.append(c)**

**elif c == ')':**

**while stack and stack[-1] != '(':**

**result.append(stack.pop())**

**stack.pop()**

**else:**

**while stack and (prec(s[i]) < prec(stack[-1]) or**

**(prec(s[i]) == prec(stack[-1]) and associativity(s[i]) == 'L')):**

**result.append(stack.pop())**

**stack.append(c)**

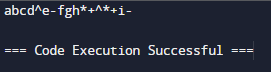
**while stack:**

**result.append(stack.pop())**

**print(''.join(result))**

**exp = "a+b\*(c^d-e)^(f+g\*h)-i"**

**infixToPostfix(exp)**



**Q2**

**class Evaluate:**

**def \_\_init\_\_(self, capacity):**

**self.top = -1**

**self.capacity = capacity**

**self.array = []**

**def isEmpty(self):**

**return True if self.top == -1 else False**

**def peek(self):**

**return self.array[-1]**

**def pop(self):**

**if not self.isEmpty():**

**self.top -= 1**

**return self.array.pop()**

**else:**

**return "$"**

**def push(self, op):**

**self.top += 1**

**self.array.append(op)**

**def evaluatePostfix(self, exp):**

**for i in exp:**

**if i.isdigit():**

**self.push(i)**

**else:**

**val1 = self.pop()**

**val2 = self.pop()**

**self.push(str(eval(val2 + i + val1)))**

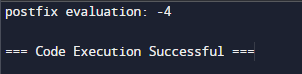
**return int(self.pop())**

**if \_\_name\_\_ == '\_\_main\_\_':**

**exp = "231\*+9-"**

**obj = Evaluate(len(exp))**

**print("postfix evaluation: %d" % (obj.evaluatePostfix(exp)))**



**Q3**

**import re**

**def evalExpr(infixStr):**

**precs = {'+':0 , '-':0, '/':1, '\*':1, '^':2}**

**funcs = {**

**'+': (lambda a,b: a+b),**

**'-': (lambda a,b: a-b),**

**'/': (lambda a,b: a/b),**

**'\*': (lambda a,b: a\*b),**

**'^': (lambda a,b: a\*\*b)**

**}**

**tokens = re.split(r' \*([\(\)\+\-\\*\^/]) \*', infixStr)**

**tokens = [t for t in tokens if t!='']**

**pos = 0**

**def eval2(minprec,closer=None):**

**nonlocal pos, tokens**

**if (pos >= len(tokens)):**

**raise Exception("Unexpected end")**

**if (tokens[pos]=="("):**

**pos += 1**

**val = eval2(0,")")**

**pos += 1**

**else:**

**val = float(tokens[pos])**

**pos += 1**

**while pos < len(tokens):**

**op = tokens[pos]**

**if op == closer:**

**return val**

**prec = precs.get(op)**

**if prec == None:**

**raise Exception("operator expected. got " + op)**

**if prec<minprec:**

**break**

**pos += 1**

**arg2 = eval2(prec+1,closer)**

**val = (funcs[op])(val, arg2)**

**if closer != None:**

**raise Exception("Expected " + closer)**

**return val**

**return eval2(0)**

**print(evalExpr("5+3\*4^2+1"))**

**print(evalExpr("7+(8/2)\*5"))**

**print(evalExpr("3\*5+8\*7"))**

