BRAC UNIVERSITY
Inspiring Excellence

Department of Computer Science and Engineering

Course Code: CSE 220	Credits: 1.5
Course Name: Data Structures	Semester: FALL’21

Lab 04

Parenthesis Balancing

I. **Topic Overview:**

Students will learn a practical use case for stack data structure. By implementing and using stacks methods, they will learn to check whether a given expression has balanced parentheses or brackets.

II. **Lesson Fit:**

For successfully completing this lab’s tasks, the students need to have a good idea on how stack is implemented and how the various methods such as push(), pop() etc works.

They also need a clear idea about array and linked list as both of these data structures are used to create a stack.

Moreover, students need a solid grasp on programming in Java and Java IDE.

III. **Learning Outcome:**

After this lecture, the students will be able to:

a. Understand the functionality of stack data structure
b. Use stack methods to solve programming problems
c. Implement stack ADT using both array and linked list

IV. **Anticipated Challenges and Possible Solutions**

a. Students might find it difficult to implement stack using array or linked list or both. For example, in array based implementation, students have to check for stack overflow and underflow exceptions. Also, in case of linked list based implementation, only stack underflow condition needs to be checked.

   Solutions:

   i. Students have to create class for stack related exceptions. Two different classes can be declared, one for overflow and one for underflow. Both classes need to extend the Exception class of Java.

   ii. Put stack's push and pop methods inside a try-catch block and handle the exceptions.

b. It can get difficult or confusing to check all conditions of parenthesis matching. Because, when a closing parenthesis/curly brace/bracket is encountered one needs to make sure the correct type of opening parenthesis/curly brace/bracket are popped from the stack

   Solutions:

   i. Students need to write if-else if checking for each type of parenthesis/curly brace/bracket separately when an opening parenthesis/curly brace/bracket is encountered.

   ii. Also, a method to check whether the stack is empty after reading all inputs needs to be implemented.

V. Acceptance and Evaluation

This task(s) will be evaluated as a lab work. So, each student is required to finish the given task within the allocated time. Partial marking may be given if someone can show decent progress in their incomplete work.

Instructions for students:

1. Complete the following problem using concepts of STACK,
2. You may use any language to complete the tasks.
3. You need to create a full functioning Stack data structure to solve this problem. You are not allowed to use the built in Stack.
4. You need to submit one single file containing all the methods/functions. The form will be open till 2 days after the due date. 30% marks will be deducted for each late day in case of late submission.
5. The submission format MUST be maintained. You need to copy paste all your codes in ONE SINGLE .txt file and upload that. If format is not maintained, whole lab submission will be canceled.
6. If you are using JAVA, you must include the main method as well which should test your other methods and print the outputs according to the tasks.
7. If you are using PYTHON, make sure your code has the methods invoked and proper printing statements according to the tasks.
8. The google form link for this lab is provided in BUX under LAB 4-Parenthesis Balancing subsection under the FALL21 CSE220 lab tab.

Lab 04 Activity List

Input

Your program will take an arithmetic expression as a String input. For Example:

1. "1+2*(3/4)"

2. "1+2*[3*3+{4–5(6(7/8/9)+10)–11+(12*8)]+14"

3. "1+2*[3*3+{4–5(6(7/8/9)+10)}–11+(12*8)/{13+13}]+14"

Program

Your program will determine whether the open brackets (the square brackets, curly braces and the parentheses) are closed in the correct order.

Outputs:

Output 1

```
1+2*(3/4)
This expression is correct.
```

Output 2

```
1+2*[3*3+{4–5(6(7/8/9)+10)–11+(12*8)]+14
This expression is NOT correct.
Error at character # 10. '{'- not closed.
```

Output 3

```
1+2*[3*3+{4–5(6(7/8/9)+10)}–11+(12*8)/{13+13}]+14
This expression is correct.
```

Output 4

```
1+2]*[3*3+{4–5(6(7/8/9)+10)–11+(12*8)]+14
This expression is NOT correct.
Error at character # 4. ‘]‘- not opened.
```

Task 1

Solve the above problem using an array based stack.

Task 2

Solve the above problem using a linked list based stack.

Assignment 04

CSE220

```
In [5]: #Task 1
        #Array Based Stack

        class stack:
            list=[]
            pointer=-1
            def push(self,element):
                self.list.append(element)
                self.pointer+=1
            def peek(self):
                if self.list==[]:
                    return "Empty List"
                else:
                    return(self.list[self.pointer])
            def pop(self):
                if self.list==[]:
                    return "Empty List"
                else:
                    value = self.list[self.pointer]
                    self.list = self.list[:-1]
                    self.pointer -= 1
                    return value
        def investigate(data):
            open=['(','{','[']
            close=[')','}',']']
            stack1=stack()
            for i in data:
                if i in open:
                    stack1.push(i)

                elif i in close:
                    if (stack1.peek()=="(" and i==")") or (stack1.peek()=="{" and i=="}") or (stack1.peek()=="[" and i=="]"):
                        stack1.pop()
                    else:
                        print("This expression is NOT correct.")
                        return

            if stack1.list==[]:
                print("This expression is correct.")
            else:
                print("This expression is NOT correct.")
```

```python
##Tester Class##
b= '1+2*(3/4)'
investigate(b)
```

```
This expression is correct.
```

In [11]:
```
#Task 2
#Linked-List Based Stack

class Node:
    def __init__(self,v,n):
        self.value=v
        self.next=n
class stack:
    head = None
    def push(self,element):
        if self.head==None:
            self.head=Node(element,None)
        else:
            n=Node(element,None)
            n.next=self.head
            self.head=n
    def peek(self):
        if self.head==None:
            return "Empty Linked-List"
        else:
            return(self.head.value)
    def pop(self):
        if self.head==None:
            return "Empty List"
        else:
            temp=self.head
            self.head=temp.next
            temp.value=None
            temp.next=None

def investigate(data):
    open=['(','{','[']
    close=[')','}',']']
    stack2=stack()
    for i in data:
        if i in open:
            stack2.push(i)
        elif i in close:
            if (stack2.peek()=="(" and i==")") or (stack2.peek()=="{" and i=="}") or (stack2.peek()=="[" and i=="]"):
                stack2.pop()
            else:
                print("This expression is NOT correct.")
                return
    if stack2.peek()== "Empty Linked-List":
        print("This expression is correct.")
    else:
        print("This expression is NOT correct.")


##Tester Class##
b= '1+2*[3*3+{4-5(6(7/8/9)+10)-11+(12*8)]+14'
investigate(b)
```

```
This expression is NOT correct.
```

```
In [ ]:
```