Activity #3 (Midterm)

1. Arithmetic expressions that may contain various pairs of grouping symbols, such as:
Parentheses: "(" and ")"
Braces: "{" and "}"
Brackets "[" and "]"
Each opening symbol must match its corresponding closing
symbol. For example, a left bracket, "[" must match a corresponding right
bracket, "]", as in the expression [(5 + x) – (y + z)]. The following examples
illustrate this concept:

Correct: ( )(( )){([( )])}
Correct: ((( )(( )){([( )])}))
Incorrect:)(( )){([( )])}
Incorrect: ({[])}
Incorrect: (

2. Create a program that reverses the lines of text of a
file using a Stack Data Structure. After executing the program, the **text file** have text which are reversed by the Stack.

ArrayStack.py code:

```
class ArrayStack:
    def __init__(self):
        self.data=[]

    def __len__(self):
        return len(self.data)

    def is_empty(self):
        return len(self)==0

    def push(self, val):
        self.data.append(val)

    def top(self):
        if self.is_empty():
            raise Exception('Stack is empty')
        return self.data[-1]

    def pop(self):
        if self.is_empty():
            raise Exception('Stack is empty')
        return self.data.pop()

    def is_balanced(expression):
        stack = ArrayStack()
        opening = "([{"
        closing = ")]}"
        pairs = {')': '(', ']': '[', '}': '{'}

        for char in expression:
            if char in opening:
                stack.push(char)
            elif char in closing:
                if stack.is_empty():
                    return False
                if stack.pop() != pairs[char]:
                    return False
```

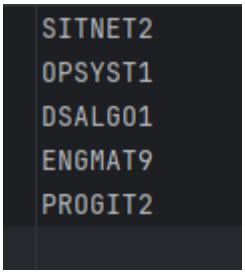
```
    return stack.is_empty()

expressions = [
    "( )(( )){([( )])}",
    "((( )(( )){([( )])})))",
    ")(( )){([( )])}",
    "({[]})",
    "("
]
```

Main.py code:

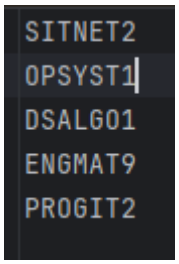
```
from ArrayStack import ArrayStack as Stack                                    Reader Mode
def is_matched(expression):
    matching_brackets = {')': '(', '}': '{', ']': '['}
    stack = []

    for char in expression:
        if char in matching_brackets.values():
            stack.append(char)
        elif char in matching_brackets.keys():
            if not stack or stack[-1] != matching_brackets[char]:
                return "the symbols are imbalanced"
            stack.pop()

    return "the symbols are balanced" if len(stack) == 0 else "the symbols are imbalanced"
user_input = input("Enter an expression to check for balanced symbols: ")
result = is_matched(user_input)
print(result)

print(is_matched("{(Okay)"))
print(is_matched("{{[Yako]}}"))
print(is_matched(")OK(( )){([( )])}"))
print(is_matched("({[]})"))
print(is_matched("("))
print(is_matched("(5x+2(3yx))"))
print()

def reverse_file(myfile):
        stack = []

        with open(myfile, 'r') as file:
            for line in file:
                stack.append(line.rstrip('\n'))

        with open(myfile, 'w') as file:
            while stack:
                file.write(stack.pop() + '\n')
```

```
myfile = 'myfile.txt'
reverse_file('myfile.txt')
print("File Contents Reversed!")
```

Myfile.txt before running:



After running:



Output:

```
Z:\DSALGO1-IDB2\Activity3_Midterms_Bullay\.venv\Scripts\python
Enter an expression to check for balanced symbols: {{([])}}
the symbols are balanced
the symbols are imbalanced
the symbols are balanced
the symbols are imbalanced
the symbols are balanced
the symbols are imbalanced
the symbols are balanced

File Contents Reversed!

Process finished with exit code 0
```