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: (

Code:

ArrayStack.py | main.py | Input.txt | Output.txt

```python
#Part 1 Code:
def validate_brackets(expression):
    stack = []
    bracket_set = {'(': ')', '{': '}', '[': ']'}
    encountered_brackets = []

    if expression and expression[0] in bracket_set.values():
        return False, expression[0]

    for char in expression:
        if char in bracket_set:
            stack.append(char)
            encountered_brackets.append(char)
        elif char in bracket_set.values():
            if not stack or bracket_set[stack.pop()] != char:
                encountered_brackets.append(char)
                return False, encountered_brackets
            encountered_brackets.append(char)

    return len(stack) == 0, encountered_brackets

user_expression = input("Enter a mathematical expression: ")

is_valid, brackets = validate_brackets(user_expression)
status_message = "Input is valid!" if is_valid else "Input is invalid!"
print(f"\n{status_message}: {''.join(brackets)}")
```

Output:

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.

Code:

```python
# Part 2 Code
def reverse_file_contents(source_path, destination_path):
    lines = []
    with open(source_path, 'r') as source_file:
        for line in source_file:
            lines.append(line.strip())

    with open(destination_path, 'w') as dest_file:
        for line in reversed(lines):
            dest_file.write(line + '\n')

output_file_path = 'Output.txt'
input_file_path = 'Input.txt'
reverse_file_contents(input_file_path, output_file_path)

print(f"Reversed lines from {input_file_path} have been saved to {output_file_path}.")
```

Output: