Stack Assignment

1. (5 pts) Here are some facts about Python's tuple data type:
   - This data type supports concatenation (`+` operator) and slicing (`[i:j]` syntax).
   - To create a tuple with a single element, syntax such as the following should be used:
     `(3,)`
     The comma is necessary.

   Given this, create a class TupleStack that implements the same functionality as ModelStack from the Stack POGIL activity but using a tuple rather than a list as the basis of the functionality. Include code (following an "`if __name__=="__main__:`" line) that demonstrates that your code is a correct implementation by creating an object of type TupleStack and calling `push()`/`pop()` a few times on this object, printing the results of the pops.

2. (5 pts) Postfix notation can be used to write arithmetic expressions without parentheses by writing operations following operands rather than between them. For instance, $2 + 3$ is written in postfix notation as $2\ 3\ +$ and $(2 + 3) * (4 + 5)$ is written in postfix notation as $2\ 3\ +\ 4\ 5\ +\ *$. Another nice feature of a postfix expression is that it's straightforward to compute its value using a stack. So, write a Python function `postfix_eval()` that accepts a string in postfix notation and returns the value of the expression represented by the string, using an ArrayStack to assist in the evaluation. For instance, `postfix_eval("23+")` should return 5 and `postfix_eval("23+45+*")` should return 45. As suggested by these examples, you can assume that each numeric value is a single digit and that the only operations are `+` and `*`. Include the following in the file in order to test your function:

```
if __name__ == "__main__":
    s1 = "23+"
    print(s1, postfix_eval(s1))
    s2 = "23+45+*"
    print(s2, postfix_eval(s2))
```