## Homework

1. Implement a method with signature transfer(S, T) that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T.

public static <T> void transfer(Stack<T> S, Stack<T> T) {

while (!S.isEmpty()) {

T.push(S.pop());

}

}

1. Give a recursive method for removing all the elements from a stack.

public static <T> void removeAll(Stack<T> stack) {

if (!stack.isEmpty()) {

stack.pop();

removeAll(stack);

}

}

1. Postfix notation is an unambiguous way of writing an arithmetic expression without parentheses. It is defined so that if “(exp1)op(exp2)” is a normal fully parenthesized expression whose operation is op, the postfix version of this is “pexp1 pexp2 op”, where pexp1 is the postfix version of exp1 and pexp2 is the postfix version of exp2. The postfix version of a single number or variable is just that number or variable. So, for example, the postfix version of “((5 + 2) ∗ (8 − 3))/4” is “5 2 + 8 3 − ∗ 4 /”. Describe a nonrecursive way of evaluating an expression in postfix notation.

public static int evaluatePostfix(String expression) {

Stack<Integer> stack = new ArrayStack<>();

String[] tokens = expression.split("\\s+"); // Split the expression into tokens

for (String token : tokens) {

if (token.matches("-?\\d+")) { // If token is a number

stack.push(Integer.parseInt(token)); // Push onto the stack

} else { // If token is an operator

int operand2 = stack.pop();

int operand1 = stack.pop();

switch (token) {

case "+":

stack.push(operand1 + operand2);

break;

case "-":

stack.push(operand1 - operand2);

break;

case "\*":

stack.push(operand1 \* operand2);

break;

case "/":

stack.push(operand1 / operand2);

break;

}

}

}

return stack.pop(); // Result is the only element left on the stack

}

1. Implement the clone( ) method for the ArrayStack class.  
   public ArrayStack<T> clone() {

try {

ArrayStack<T> clonedStack = (ArrayStack<T>) super.clone();

clonedStack.elements = Arrays.copyOf(elements, size);

return clonedStack;

} catch (CloneNotSupportedException e) {

throw new Error("This should never happen");

}

}

1. Implement a program that can input an expression in postfix notation (see Exercise C-6.19) and output its value

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an expression in postfix notation: ");

String expression = scanner.nextLine();

int result = evaluatePostfix(expression);

System.out.println("Result: " + result);

}