COSC2436: Exam 2 Version 1 Key

Stacks Question #1 (10 points)

- Initialize and use stack<char> - 1 point
- Add digits to string - 1 point
- Push ‘(‘ onto stack - 1 point
- When ‘)’ is found, empty stack into string until stack.top() == ‘(‘ - 1 points
- If operator:
  - While ***!stack.empty() && priority(s[i]) <= priority(stack.top())*** - 1 point
  - Add top of stack to string - 1 point
  - Pop from stack (to prevent infinite loop) - 1 point
  - Push operator onto stack - 1 points
- Empty stack into string - 1 point
- Return postfix string - 1 point

Stacks Question #1

```
20 ▼ string infixToPostfix(string exp, int size){
21     stack<char> s;
22     string str;
23 ▼   for(int i = 0; i < size; i++){
24 ▼     if(isdigit(exp[i])){
25         str += exp[i];
26       }
27 ▼     else if(exp[i] == '('){
28         s.push('(');
29       }
30 ▼     else if(exp[i] == ')'){
31 ▼       while(s.top() != '('){
32           str += s.top();
33           s.pop();
34         }
35         s.pop();
36       }
37 ▼     else{
38 ▼       while(!s.empty() && priority(exp[i]) <= priority(s.top())){
39           str += s.top();
40           s.pop();
41         }
42         s.push(exp[i]);
43       }
44     }
45 ▼   while(!s.empty()){
46       str += s.top();
47       s.pop();
48     }
49     return str;
50 }
```

Stacks Question #2 (5 points)

- Initialize and use stack<int> - 1 point
- Check for digits and push them to stack - 1 point
- Check for operator - 1 point
- Correct order (val2 before val1) - 1 point
- Return an int value - 1 point

Stacks Question #2

```
183 ▼ int evalPostfix(string s) {
184     stack<int> st;
185 ▼   for(int i = 0; i < s.length(); i ++) {
186 ▼     if(isdigit(s[i])) {
187         st.push(s[i] - 48);
188       }
189 ▼     else {
190         int val1 = st.top(); st.pop();
191         int val2 = st.top(); st.pop();
192 ▼       switch(s[i]) {
193           case '+': st.push(val2 + val1); break;
194           case '-': st.push(val2 - val1); break;
195           case '*': st.push(val2 * val1); break;
196           case '/': st.push(val2 / val1); break;
197         }
198       }
199   }
200   return st.top();
201 }
```

Stacks Question #3 (5 points)

Convert **623/+472-*-** from postfix to infix

- Correct Answer - 5 points:

((6+(2/3))-(4*(7-2))) or **(6+(2/3))-(4*(7-2))** or **6+(2/3)-4*(7-2)** or **6+2/3-4*(7-2)**

- Wrong Answer - 4 points:

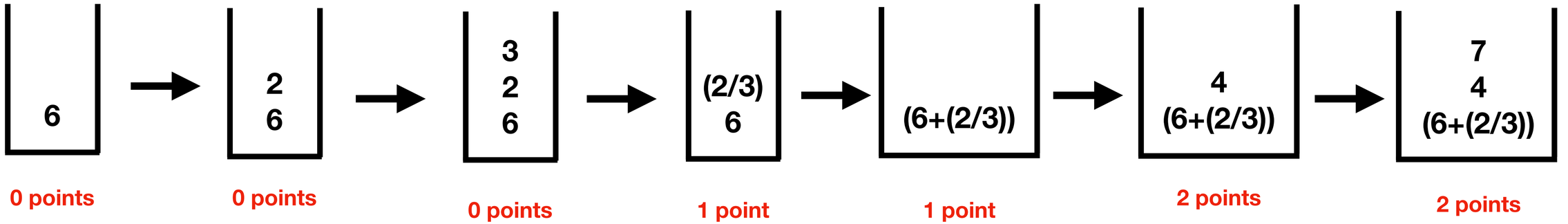
6+2/3-4*7-2

- Correct Tracing - points vary

- No/Wrong Tracing or Wrong Answer - 0 points

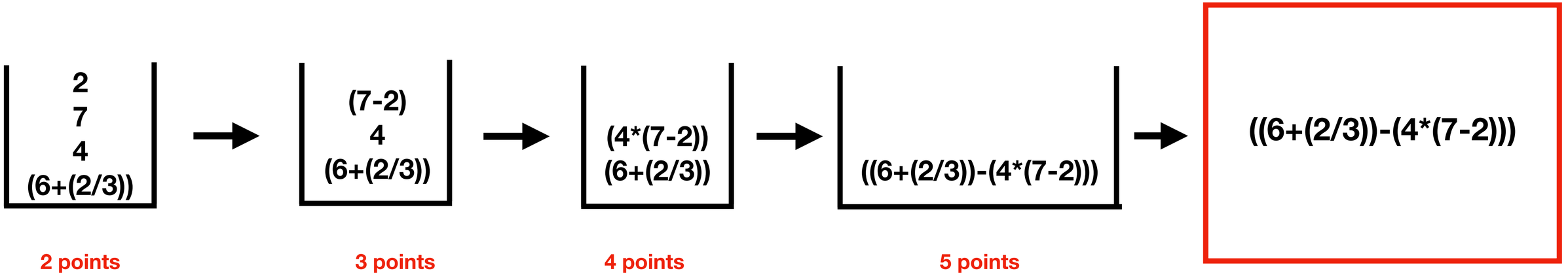
Stacks Question #3

The points below each trace indicate how many points will be received if they stopped at that point.



0 points | 0 points | 0 points | 1 point | 1 point | 2 points | 2 points

Final Answer:



2 points | 3 points | 4 points | 5 points | 5 points

Stacks Question #4 (5 points)

Convert **((5+2)*8)/6+3*(6-4)** from infix to postfix

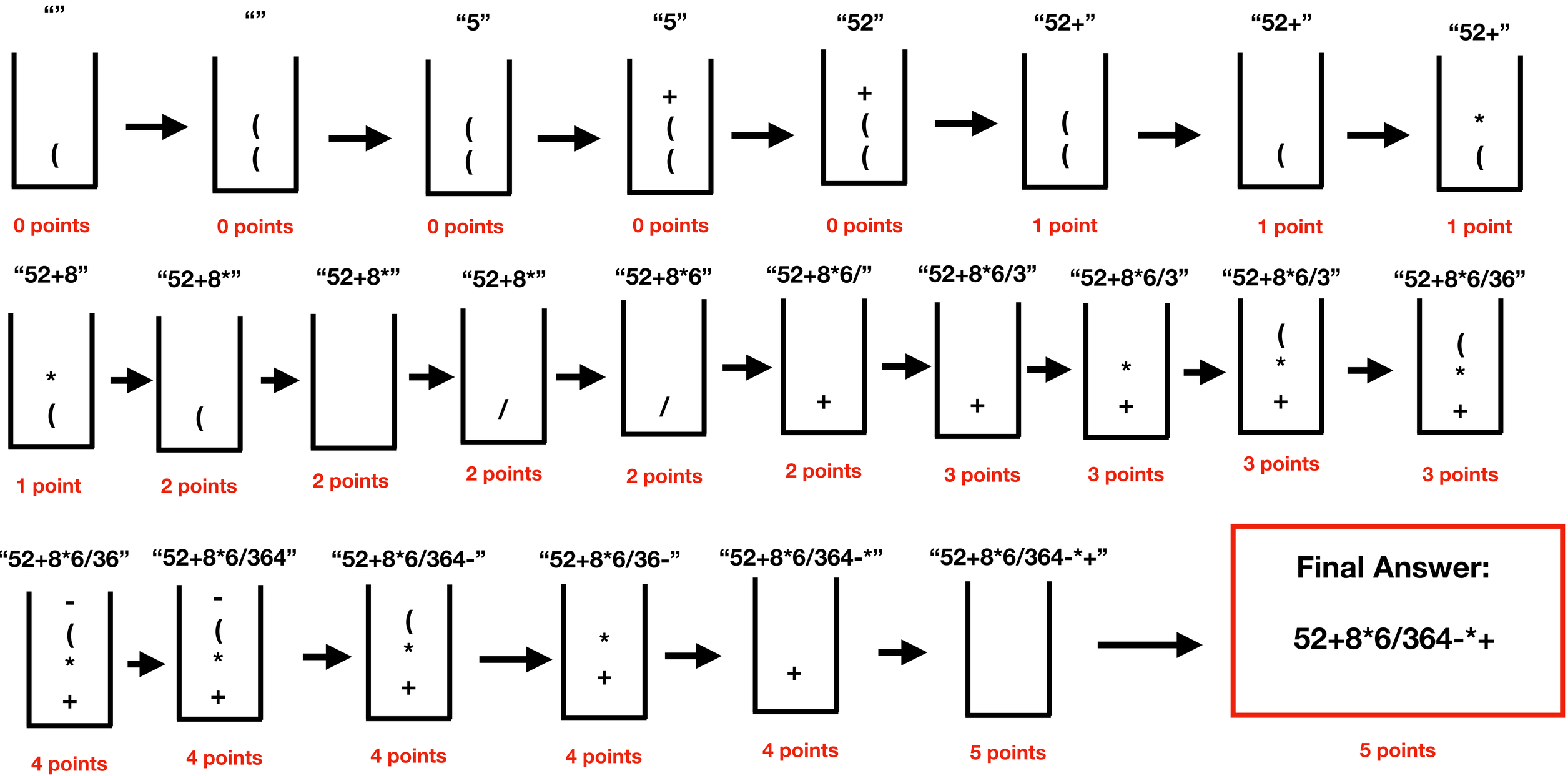
- Correct Answer - 5 points:

  52+8*6/364-*+

- Correct Tracing - points vary

- No/Wrong Tracing or No Answer - 0 points

Stacks Question #4 (The points below the trace indicate how many points will be received if trace was ended there)



COSC2436: Exam 2 Version 2 Key

Stacks Question #1 (10 points)

- Initialize and use stack<char> - 1 point
- Add digits to string - 1 point
- Push ‘(‘ onto stack - 1 point
- When ‘)’ is found, empty stack into string until stack.top() == ‘(‘ - 1 points
- If operator:
  - While ***!stack.empty() && priority(s[i]) <= priority(stack.top())*** - 1 point
  - Add top of stack to string - 1 point
  - Pop from stack (to prevent infinite loop) - 1 point
  - Push operator onto stack - 1 points
- Empty stack into string - 1 point
- Return postfix string - 1 point

Stacks Question #1

```
20 ▼ string infixToPostfix(string exp, int size){
21     stack<char> s;
22     string str;
23 ▼   for(int i = 0; i < size; i++){
24 ▼     if(isdigit(exp[i])){
25         str += exp[i];
26       }
27 ▼     else if(exp[i] == '('){
28         s.push('(');
29       }
30 ▼     else if(exp[i] == ')'){
31 ▼       while(s.top() != '('){
32           str += s.top();
33           s.pop();
34         }
35         s.pop();
36       }
37 ▼     else{
38 ▼       while(!s.empty() && priority(exp[i]) <= priority(s.top())){
39           str += s.top();
40           s.pop();
41         }
42         s.push(exp[i]);
43       }
44     }
45 ▼   while(!s.empty()){
46       str += s.top();
47       s.pop();
48     }
49     return str;
50 }
```

Stacks Question #2 (5 points)

- Push open parenthesis/brackets into stack - 1 point
- If/else if statements for all closing parenthesis/brackets - 1 point
- If stack is empty or top does not correspond with closing parenthesis/ bracket - 1 point
- Pop from stack after checking to see if closing parenthesis match one parenthesis - 1 point
- Check if stack is empty at the end - 1 point

Stacks Question #2

```
10 bool validParenthesis(string exp){
11   stack<char> st;
12   for(int i = 0; i < exp.length(); i++){
13     if(exp[i] == '(' || exp[i] == '[' || exp[i] == '{'){
14       st.push(exp[i]);
15     }
16     else if(exp[i] == ')'){
17       if(st.empty() || st.top() != '('){
18         return false;
19       }
20       st.pop();
21     }
22     else if(exp[i] == ']'){
23       if(st.empty() || st.top() != '['){
24         return false;
25       }
26       st.pop();
27     }
28     else if(exp[i] == '}'){
29       if(st.empty() || st.top() != '{'){
30         return false;
31       }
32       st.pop();
33     }
34   }
35   return st.empty();
36 }
```

Stacks Question #3 (5 points)

Convert **94-78/53+*+** from postfix to infix

- Correct Answer - 5 points:

((9-4)+((7/8)*(5+3))) or **(9-4)+((7/8)*(5+3))** or **(9-4)+(7/8)*(5+3)** or **9-4+(7/8)*(5+3)** or

9-4+7/8*(5+3)

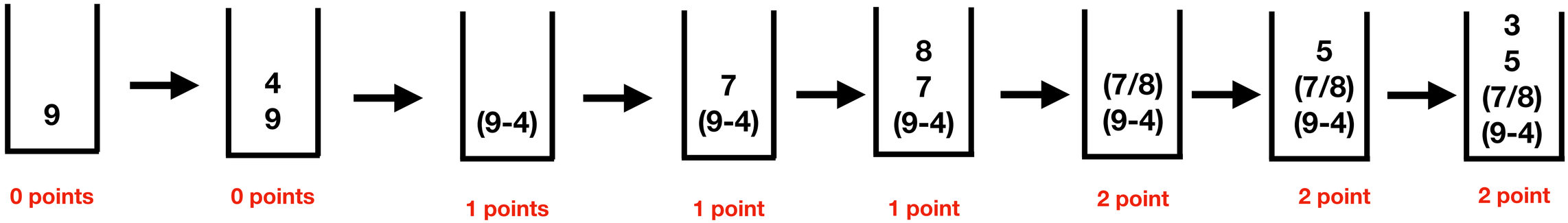
- Wrong Answer - 4 points:

9-4+7/8*5+3

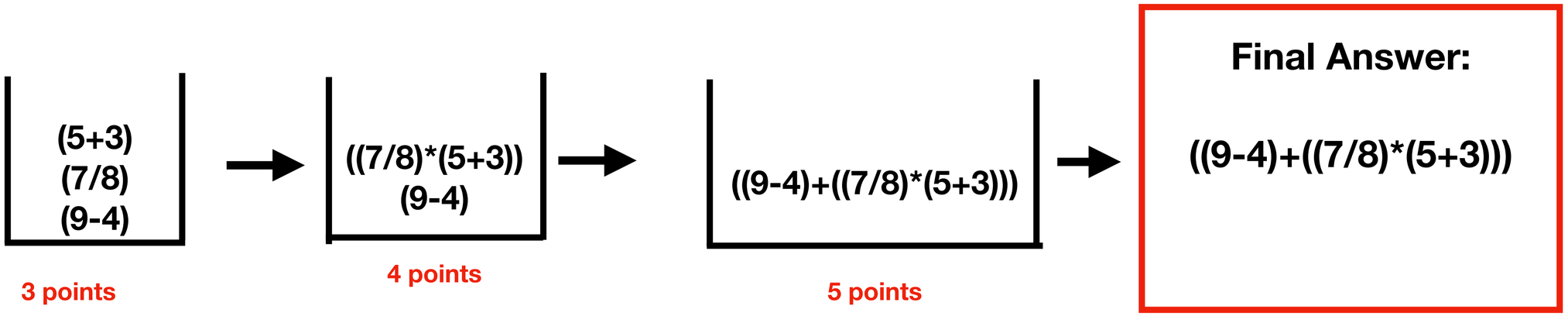
- Correct Tracing - points vary
- No/Wrong Tracing or Wrong Answer - 0 points

Stacks Question #3

The points below each trace indicate how many points will be received if they stopped at that point.



0 points | 0 points | 1 points | 1 point | 1 point | 2 point | 2 point | 2 point



3 points | 4 points | 5 points | 5 points

Stacks Question #4 (5 points)

Convert **(1+(8-2))-6/3*(5/7)** from infix to postfix

- Correct Answer - 5 points:

  182-+63/57/*-

- Correct Tracing - points vary

- No/Wrong Tracing or No Answer - 0 points

Stacks Question #4 (The points below the trace indicate how many points will be received if trace was ended there)