Junyu Teoh
CSCD300 HW56 Progress

(The following answers are based on the algorithm Tony Tian discussed in class, specifically R04_Stack3.pdf on canvas. I will not copy the code here, but all of the explanation here is based on that algorithm.)

Question 1: (((1+2)-(3-4))/(6-5)) to postfix.

Initially, the stack is empty. We create an empty string variable named outputString.

1st iteration: The current item is "(", we push it on the stack. Stack currently only has "(", and top points to "(". outputString is empty. Move onto the next item.

2nd iteration: The current item is "(", we push it on the stack. Stack now has "(", "(". Top points to the latest entry of "(". outputString is empty. Move onto the next item.

3rd iteration: The current item is "(", we push it on the stack. Stack now has "(", "(", "(". Top points to the latest entry of "(". outputString is empty. Move onto the next item.

4th iteration: The current item is 1, an operand. It satisfies the if control block, so we write it to outputString. Stack = "(", "(", "(". Top points to the latest entry of "(". outputString = 1. Move onto the next item.

5th iteration: The current item is +, an operator. We do the else block. Stack is not empty but the precedence of stack's top is lower than the current item, thus we fall out of that block. We push the current item onto the stack. Stack = "(", "(", "(", "+". Top points to "+". outputString = 1. Move onto the next item.

6th iteration: The current item is 2, an operand. It satisfies the if control block, so we write it to outputString. Stack = "(", "(", "(", "+". Top points to "+". outputString = 1 2. Move onto the next item.

7th iteration: The current item is ")". We do the else if condition. The item on the top of the stack is not "(", so we pop the stack and write it to the end of our outputString. Then, we pop "(" and discard it. Stack = "(", "(". Top points to "(". outputString = 1 2 +. Move onto the next item.

8th iteration: The current item is "-", an operator. We do the else block. Stack is not empty, but the precedence of the stack's top is lower than the current item, thus we fall out of that block. We push the current item onto the stack. Stack = "(", "(", "-". Top points to "-". outputString = 1 2 +. Move onto the next item.

9th iteration: The current item is "(". We push it onto the stack. Stack = "(", "(", "-", "(". Top points to "(". outputString = 1 2 +. Move onto the next item.

10th iteration: The current item is 3, an operand. It satisfies the control block, so we write it to outputString. Stack = "(", "(", "-", "(". Top points to "(". outputString = 1 2 + 3. Move onto the next item.

11th iteration: The current item is "-", an operator. We do the else block. Stack is not empty, but the precedence of the stack's top is lower than the current item, thus we fall out of that block. We push the current item onto the stack. Stack = "(", "(", "-", "(", "-". Top points to "-". outputString = 1 2 + 3. Move onto the next item.

Junyu Teoh
CSCD300 HW56 Progress

12th iteration: The current item is 4, an operand. It satisfies the control block, so we write it to outputString. Stack = "(", "(", "-", "(", "-". Top points to "-". outputString = 1 2 + 3 4. Move onto the next item.

13th iteration: The current item is ")". We do the else if condition. The item on the top os the stack is not "(", so we pop the stack and write it to the end of our outputString. Then we pop "(" and discard it. Stack = "(", "(", "-".Top points to "-". outputString = 1 2 + 3 4 -. Move onto the next item.

14th iteration: The current item is ")". We do the else if condition. The item on the top of the stack is not "(", so we pop the stack and write it to the end of our outputString. Then we pop "(" and discard it. Stack = "(". Top points to "(". outputString = 1 2 + 3 4 - -. Move onto the next item.

15th iteration: The current item is /, an operator. We do the else block. Stack is not empty, but the precedence of the stack's top is lower than the current item, thus we fall out of that block. We push the current item onto the stack. Stack = "(", "/". Top points to "/". outputString = 1 2 + 3 4 - -. Move onto the next item.

16th iteration: The current item is "(". We push it onto the stack. Stack = "(", "/", "(". Top points to "(". outputString = 1 2 + 3 4 - -. Move onto the next item.

17th iteration: The current item is 6, an operand. It satisfies the control block, so we write it to outputString. Stack = "(", "/", "(". Top points to "(". outputString = 1 2 + 3 4 - - 6. Move onto the next item.

18th iteration: The current item is "-", an operator. We do the else block. Stack is not empty, but the precedence of the stack's top is lower than the current item, thus we fall out of that block. We push the current item onto the stack. Stack = "(", "/", "(", "-". Top points to "-". outputString = 1 2 + 3 4 - - 6. Move onto the next item.

19th iteration: The current item is 5, and operand. It satisfies the control block, so we write it to outputString. Stack = "(", "/", "(", "-". Top points to "-". outputString = 1 2 + 3 4 - - 6 5. Move onto the next item.

20th iteration: The current item is ")". We do the else if condition. The item on the top of the stack is not "(", so we pop the stack and write it to the end of our outputString. Then, we pop "(" and discard it. Stack = "(", "/".Top points to "/". outputString = 1 2 + 3 4 - - 6 5 -. Move onto the next item.

21st iteration: : The current item is ")". We do the else if condition. The item on the top of the stack is not "(", so we pop the stack and write it to the end of our outputString. Then, we pop "(" and discard it. Stack is empty. Top points to null. outputString = 1 2 + 3 4 - - 6 5 - /. End.

Junyu Teoh
CSCD300 HW56 Progress

Question 2: “1 2 + 3 4 - - 6 5 - /” to infix and evaluate

We first create an empty stack.

1st iteration: Current item is 1, an operand. If statement is true, we push onto the stack. Stack = 1. Top points to 1. Move onto the next item.

2nd iteration: Current item is 2, an operand. If statement is true, we push onto the stack. Stack = 1, 2. Top points to 2. Move onto the next item.

3rd iteration: Current item is +, an operator. Do else. Pop stack once, attach it to the right of the operator, then pop again and attach it to the left. We have 1+2. Evaluate it to get 3, and push it back onto the stack. Stack = 3. Top points to 3. Move onto the next item.

4th iteration: Current item is 3, an operand. If statement is true, we push onto the stack. Stack = 3, 3. Top points to 3. Move onto the next item.

5th iteration: Current item is 4, an operand. If statement is true, we push onto the stack. Stack = 3, 3, 4. Top points to 4. Move onto the next item.

6th iteration: Current item is -, an operator. Do else. Pop stack once, attach it to the right of the operator, then pop again and attach it to the left. We have 3-4. Evaluate it to get -1, and push it back onto the stack. Stack = 3, -1. Top points to -1. Move onto the next item.

7th iteration: Current item is -, an operator. Do else. Pop stack once, attach it to the right of the operator, then pop it again and attach it to the left. We have 3 – (-1). Evaluate it to get 4, and push it back onto the stack. Stack = 4. Top points to 4.Move onto the next item.

8th iteration: Current item is 6, an operand. If statement is true, we push onto the stack. Stack = 4, 6. Top points to 6. Move onto the next item.

9th iteration: Current item is 5, an operand. If statement is true, we push onto the stack. Stack = 4, 6, 5. Top points to 5. Move onto the next item.

10th iteration: Current item is -, an operator. Do else. Pop stack once, attach it to the right of the operator, then pop it again and attach to the left. We have 6-5. Evaluate it to get 1, and push it back onto the stack. Stack = 4, 1. Top points to 1. Move onto the next item.

11th iteration: Current item is /, an operator. Do else. Pop stack once, attach it to the right of the operator, then pop it again and attach to the left. We have 4/1. Evaluate it and get 4, and push it back onto the stack. Stack =4. Top points to 4. No more item.

12th iteration: There are no more items in the post fix expression and there is only one item left on the stack. Pop to get final result. Final result is 4.