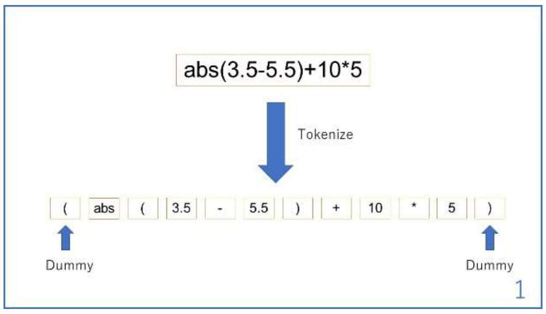
Calculator

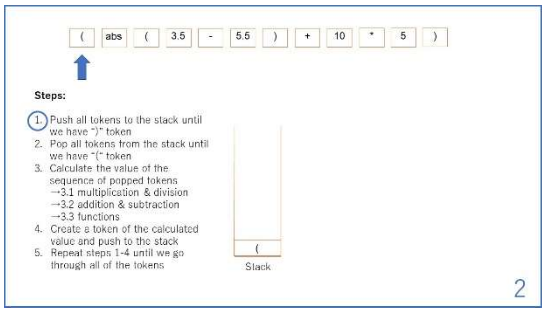
Naho Yokoyama

Overview

This program implements a calculator. When a user inputs a mathematical expression, the program will calculate and returns its value. This program supports calculation including addition, subtraction, multiplication, division, the use of parenthesis, absolute value function, integer value function, and round function.

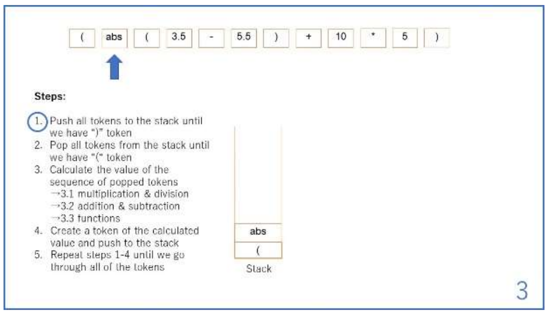
How this program works





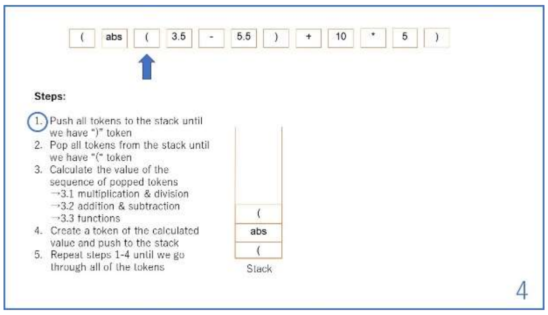
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



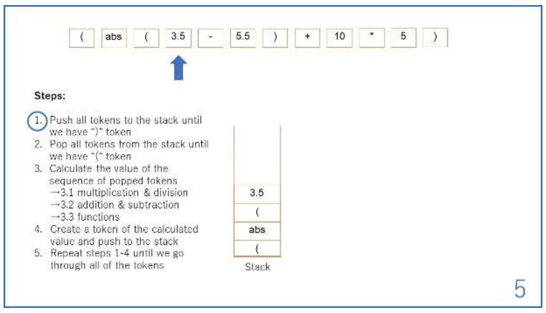
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



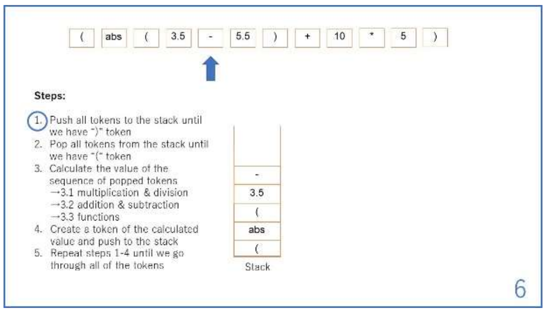
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



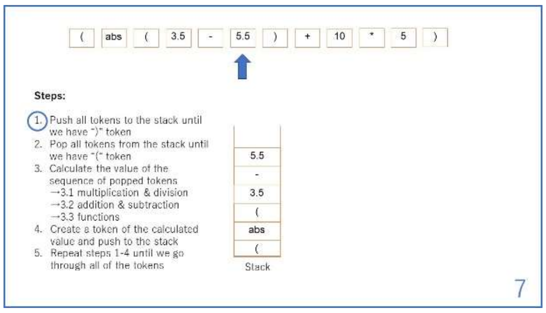
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



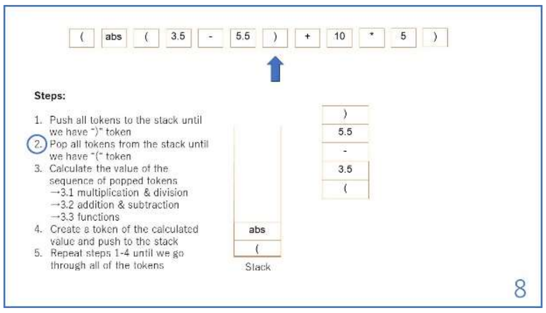
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



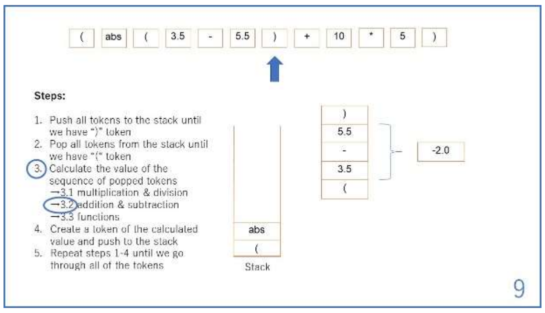
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens



Steps:

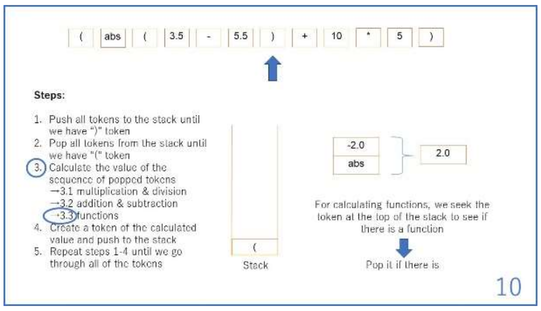
1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

9


(abs (3.5 - 5.5) + 10 * 5)

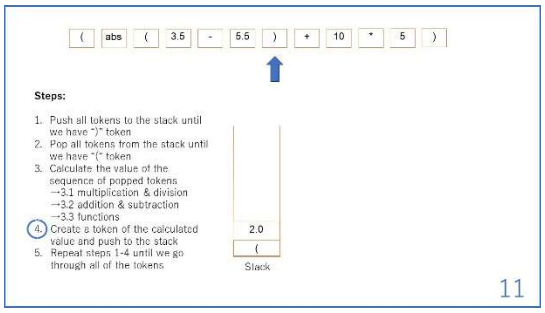
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

For calculating functions, we seek the token at the top of the stack to see if there is a function

Pop it if there is

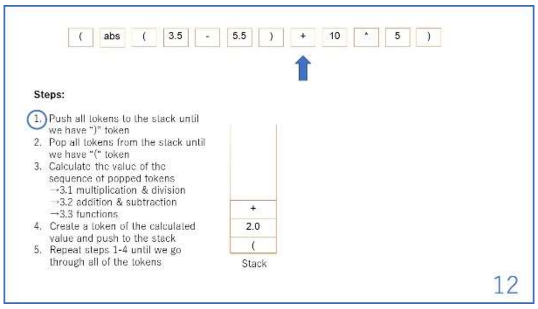
10


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

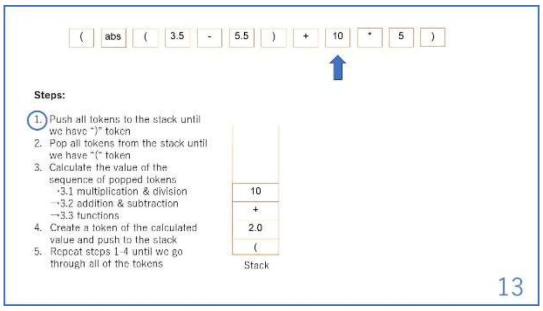
11


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

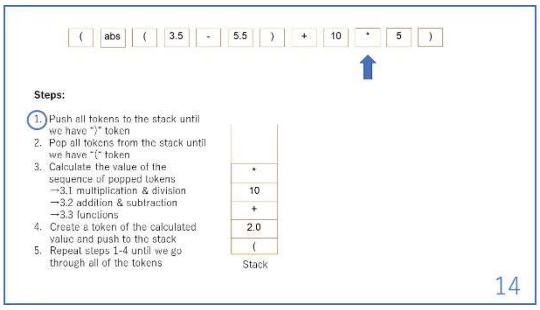
12


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

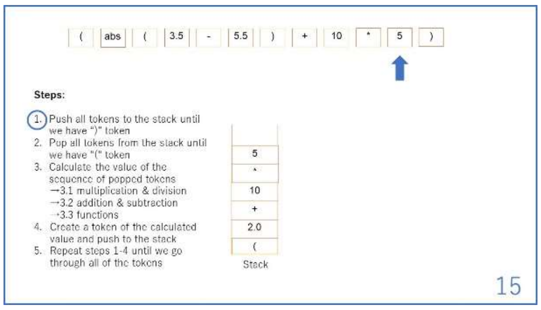
13


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

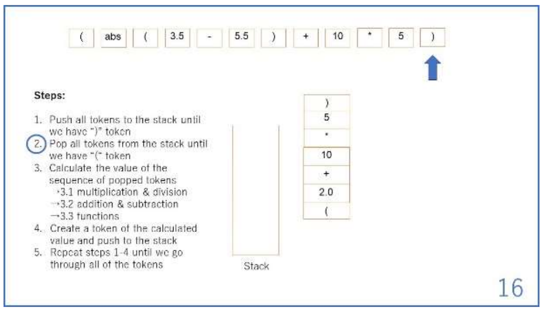
14


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

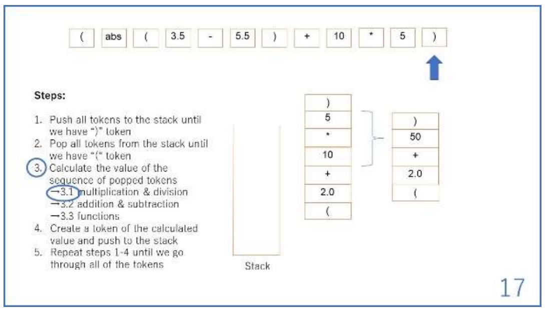
15


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

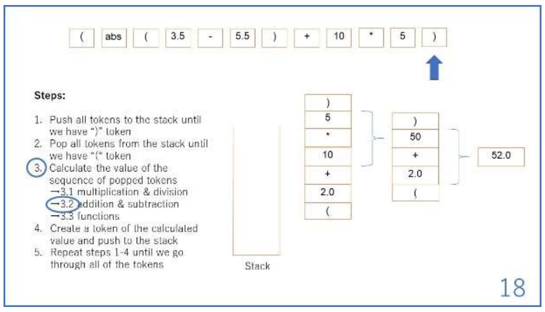
16


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

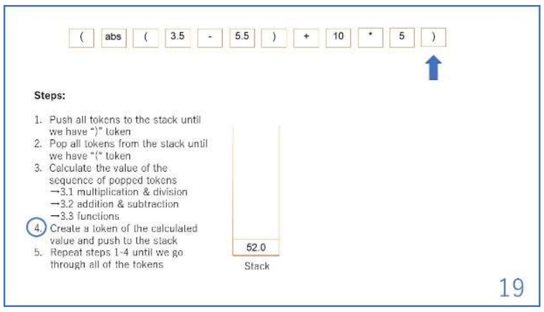
17


(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
→3.1 multiplication & division
→3.2 addition & subtraction
→3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

18



(abs (3.5 - 5.5) + 10 * 5)

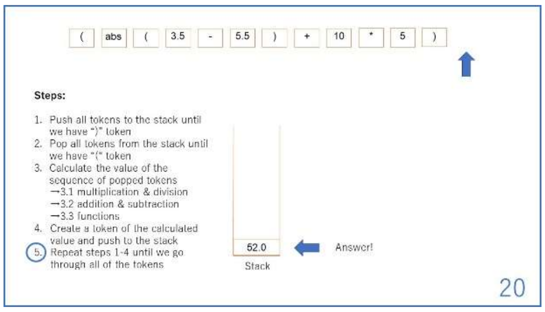
Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

52.0

Stack

19



(abs (3.5 - 5.5) + 10 * 5)

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
   →3.1 multiplication & division
   →3.2 addition & subtraction
   →3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

52.0

Answer!

Stack

20

Input's tolerance

Allow

・negative numbers (ex. "-1+-5" will be evaluated as -6)

・use of more parenthesis than necessary (ex. "(((((5+5)))))")

・too many open parenthesis (outer parenthesis will be ignored, ex. "(3+5*2" becomes 13)

　→"eval" function to calculated expected answer will returns an error though

Not allow

・division by zero

・space between characters (ex. "1 + 5")

・incomplete input of decimals (ex. ".5")

・Too many close parenthesis (ex. "3+5)")