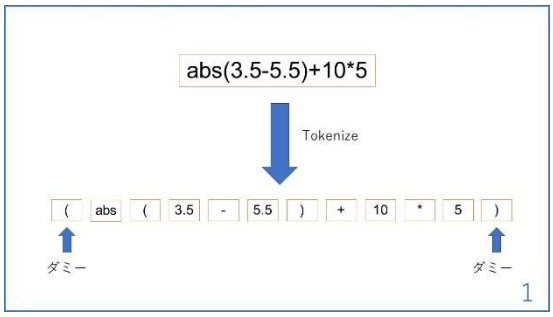
電卓プログラム

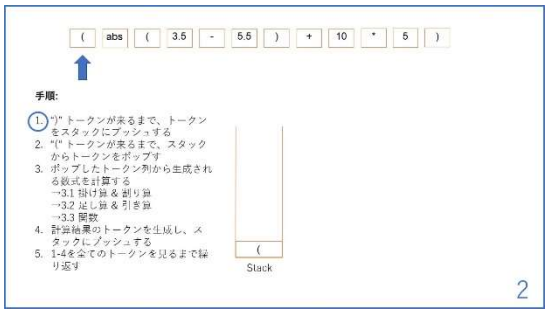
横山奈穂

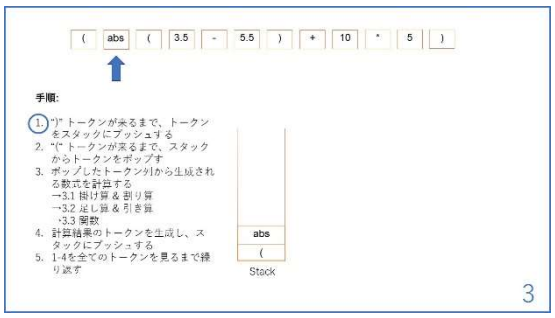
概要

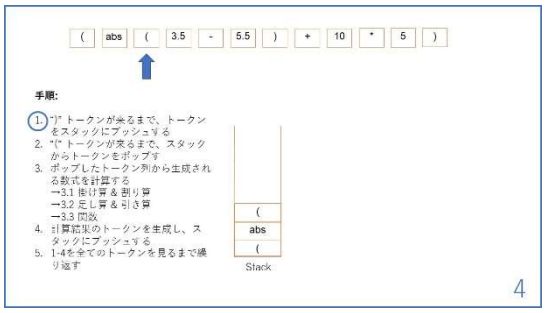
このプログラムは電卓を実装しています。ユーザーが数式を入力すると、プログラムはその値を計算して返します。対象とする演算は、足し算、引き算、掛け算、割り算、括弧の使用、絶対値関数、整数値関数、および四捨五入関数です。

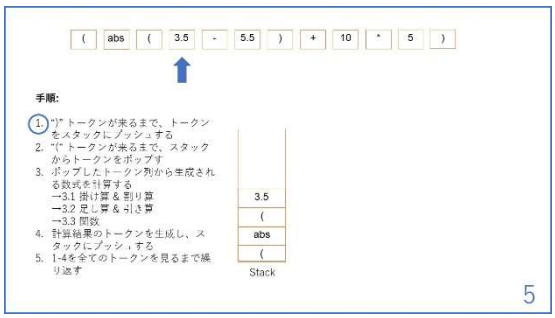
プログラムの仕組み

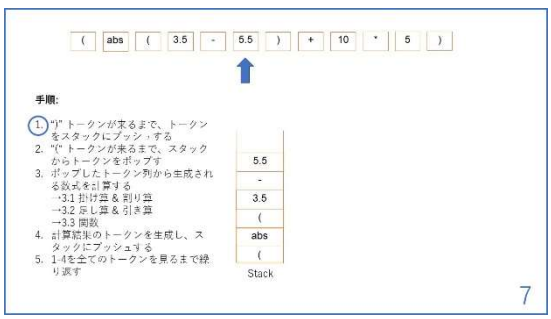


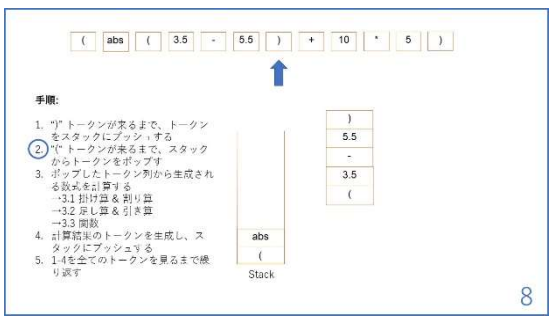


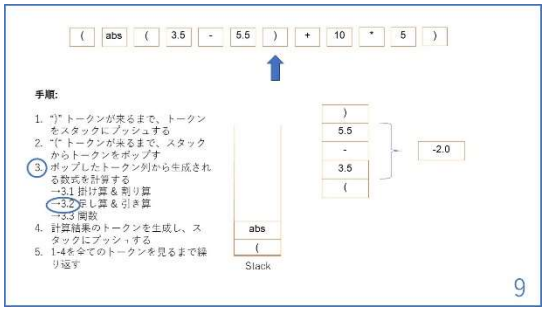


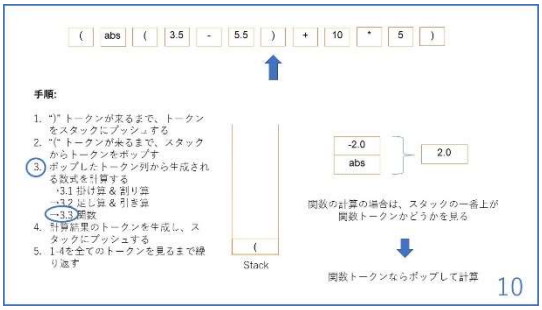


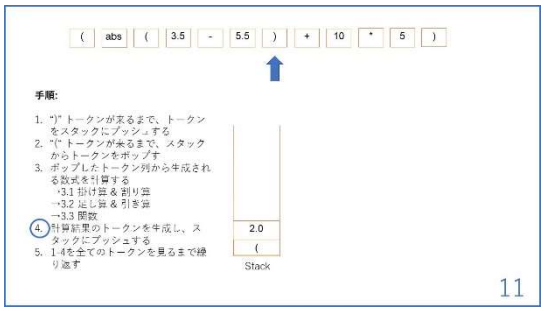


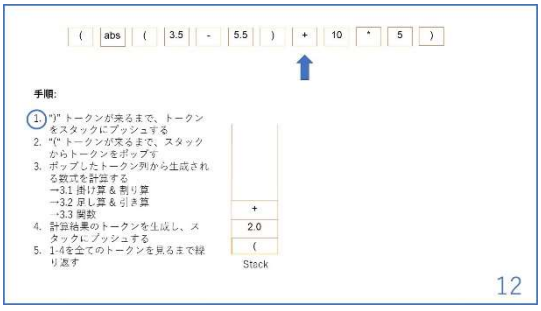


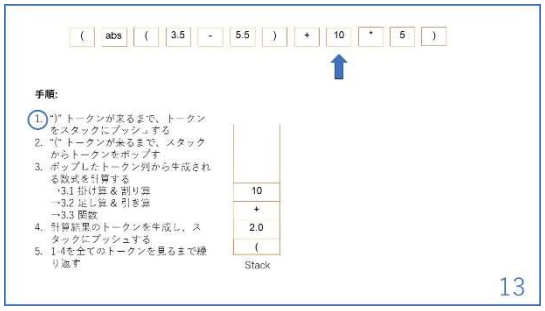


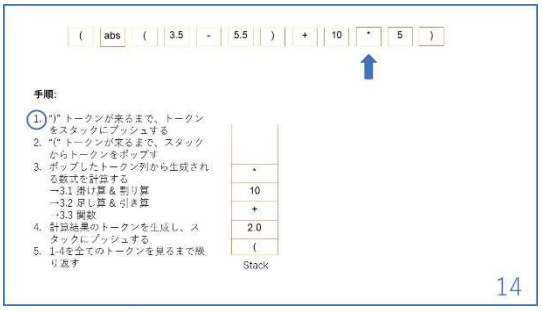

(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
abs
(
Stack
)
5.5
-
3.5
(
-2.0
9

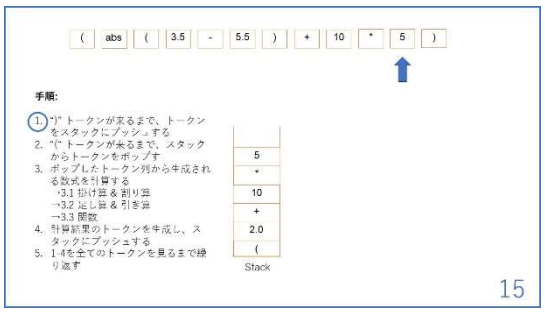

(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
(
Stack
-2.0
abs
2.0
関数の計算の場合は、スタックの一番上が関数トークンかどうかを見る
関数トークンならポップして計算
10


(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
2.0
(
Stack
11

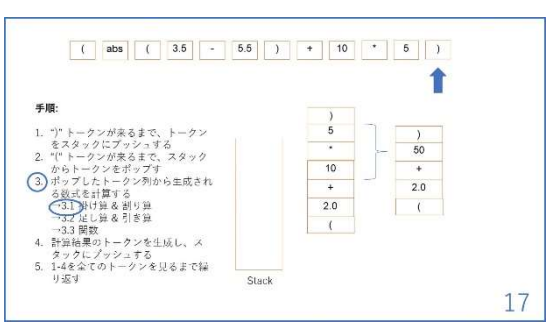

(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
+
2.0
(
Stack
12

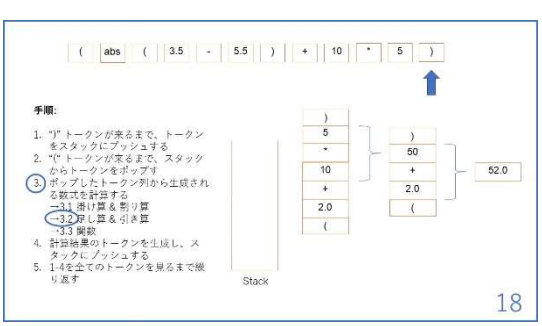

(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
10
+
2.0
(
Stack
13

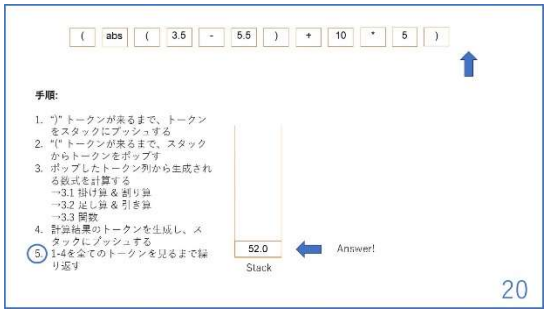

(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
*
10
+
2.0
(
Stack
14


(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
5
+
10
+
2.0
(
Stack
15


(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
)
5
*
10
+
2.0
(
Stack
16


(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
)
5
*
10
+
2.0
(
)
50
+
2.0
(
Stack
17


(abs (3.5 - 5.5) + 10 * 5)
手順:
1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップす
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算＆割り算
→3.2 足し算＆引き算
→3.3 関数
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す
)
5
*
10
+
2.0
(
)
50
+
2.0
(
52.0
Stack
18



入力の対象とするもの･しないもの

対象

・負の値（例. “-1+-5”は-6 と計算される）

・必要以上に括弧のペアを使用する(例. “(((((5+5)))))”の計算結果は”5+5”と同様)

・括弧のペアのうち、”(“を多く使用する（外側の括弧が無視される、例. “(3+5*2”は”3+5*2” となり答えは 13 となる）

　→ただし、電卓プログラムが正しい答えを返すか判定する”eval” 関数はエラーを返す

対象ではない

・ゼロ除算

・数式中に空白を入れる（例. “1 + 5”）

・不完全な数の入力(例. “.5”)

・括弧のペアのうち、”)”を多く使用する（例. “3+5)”）