GIT Department of Computer Engineering CSE 222/505 - Spring 2020 Homework #4 Part 1 Report

Murat YILDIZ 1801042004

Q1:

Convert the infix expressions given below to prefix and postfix, then evaluate them. Show your work (how each token is processed and content of the stack afterwards) step by step.

i)

Infix to postfix

CONVERSION

| Expression | Action | Stack | Effect on postfix |
|--------------------------------------|--|------------------|-------------------|
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append A to postfix | | Α |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher, Push + | + | А |
| A + ((B - C * D) / E) + F - G / H | Push (| + | А |
| A+((B-C*D)/E)+F-G/H | Push (| ((+ | А |
| A + ((B - C * D)/E)+F-G/H | It is an operand, Append B to postfix | ((+ | АВ |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher , Push – | - ((+ | АВ |
| A+((B-C*D)/E)+F-G/H | It is an operand, Append C to postfix | - ((+ | A B C |
| A+((B-C*D)/E)+F-G/H | It is an operator, Its precedence is higher , Push * | * | ABC |

| | T | | , |
|--------------------------------------|---|-----------------------|---|
| | | + | |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append D to postfix | * - ((+ | ABCD |
| A + ((B - C * D) / E) + F - G / H | Pop and append to postfix until precedence is equal or less than) | (+ | A B C D * - |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher, Push/ | (+ | A B C D * - |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append E to postfix | / (+ | A B C D * – E |
| A + ((B - C * D) / E) + F - G / H | Pop and append to postfix until precedence is equal or less than) | + | A B C D * – E / |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher, append to postfix | + | A B C D * - E / + |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append F to postfix | + | A B C D * - E / + F |
| A + ((B - C * D) / E) + F - G / H | Pop and append to postfix until precedence is equal or less than – and Push – | _ | A B C D * - E / + F + |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append G to postfix | _ | A B C D * - E / + F + G |
| A + ((B - C * D) / E) + F - G / H | Pop and append to postfix until precedence is equal or less than / and Push / | _ | A B C D * - E / + F + G |

| A+((B-C*D)/E)+F-G/H | It is an operand, Append H to postfix | / | A B C D * - E / + F + G H |
|---------------------|---|---|-----------------------------|
| A+((B-C*D)/E)+F-G/H | When it is ended Pop and append to postfix until stack is empty | _ | A B C D * - E / + F + G H / |
| A+((B-C*D)/E)+F-G/H | When it is ended Pop and append to postfix until stack is empty | | ABCD*-E/+F+GH/- |

i)

POSTFIX

EVALUATION

SCANNING LEFT TO RIGHT

| Expression | Action | Stack |
|-------------------------------|--|--------------------|
| A B C D * - E / + F + G H / - | Push A | 20 |
| A B C D * - E / + F + G H / - | Push B | 15 20 |
| A B C D * - E / + F + G H / - | Push C | 2 15 20 |
| ABCD*-E/+F+GH/- | Push D | 6 2 15 20 |
| A B C D * - E / + F + G H / - | Pop twice , Do math and push result to stack | 12 15 20 |
| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 3 20 |
| A B C D * - E / + F + G H / - | Push E | 3 3 20 |

| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 20 |
|-------------------------------|---|-----------------|
| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 21 |
| A B C D * - E / + F + G H / - | Push F | 8 21 |
| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 29 |
| A B C D * - E / + F + G H / - | Push G | 16 29 |
| A B C D * - E / + F + G H / - | Push H | 4 16 29 |
| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 4 29 |
| A B C D * - E / + F + G H / - | Pop twice, Do math and push result to stack | 25 |

i)

Infix to prefix

CONVERSION

SCANNING RIGHT TO LEFT

| Expression | Action | Stack | Effect on prefix |
|--------------------------------------|--|-------|------------------|
| A + ((B - C * D) / E) + F - G / H | It is an operand, | | Н |
| | Append H to prefix | | |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher , Push / | / | Н |

| A + ((B - C * D) / E) + F - G / H | It is an operand, Append G to prefix | / | HG |
|--------------------------------------|---|-----------------------|-----------|
| A + ((B – C * D) / E) + F – G / H | Pop and append to prefix until precedence is equal or less than – and Push – | _ | HG/ |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append G to prefix | _ | HG/F |
| A + ((B - C * D) / E) + F - G / H | Pop and append to prefix until precedence is equal or less than + and Push + | + | HG/F- |
| A + ((B – C * D) / E) + F – G / H | Push) |) + | HG/F- |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append E to prefix |) + | HG/F-E |
| A + ((B - C * D) / E) + F - G / H | It is an operator, Its precedence is higher ,Push / |) + | HG/F-E |
| A + ((B - C * D) / E) + F - G / H | Push) |) /) + | HG/F-E |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append D to prefix |) /) + | HG/F-ED |
| A + ((B - C * D) / E) + F - G / H | It is an operator, its precedence is higher ,Push * | *) /) + | HG/F-ED |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append C to prefix | *) /) + | HG/F-EDC |
| A + ((B - C * D) / E) + F - G / H | Pop and append to prefix until precedence is equal or less than – and Push – | -) /) + | HG/F-EDC* |

| A+((B-C*D)/E)+F-G/H | It is an operand, Append B to prefix | -) /) + | HG/F-EDC*B |
|--------------------------------------|---|-----------------------|-----------------|
| A+((B-C*D)/E)+F-G/H | Pop and append to prefix until precedence is equal or less than (|) + | HG/F-EDC*B- |
| A + ((B - C * D) / E) + F - G / H | Pop and append to prefix until precedence is equal or less than (| + | HG/F-EDC*B-/ |
| A+((B-C*D)/E)+F-G/H | Pop and append to prefix until precedence is equal or less than + and Push + | + | HG/F-EDC*B-/+ |
| A + ((B - C * D) / E) + F - G / H | It is an operand, Append A to prefix | + | HG/F-EDC*B-/+A |
| A+((B-C*D)/E)+F-G/H | When it is ended Pop and append to prefix until stack is empty | | HG/F-EDC*B-/+A+ |

We reverse the output string to get prefix expression so;

i)

PREFIX

EVALUATION

SCANNING RIGHT TO LEFT

A = 20, B = 15, C = 2, D = 6, E = 3, F = 8, G = 16, H = 4

| Expression | Action | Stack |
|-------------------------------|--|-------|
| + A + / - B * C D E - F / G H | Push H | 4 |
| + A + / - B * C D E - F / G H | Push G | 16 4 |
| + A + / - B * C D E - F / G H | Pop twice , Do math and push result to stack | 4 |
| + A + / - B * C D E - F / G H | Push F | 8 4 |

| + A + / - B * C D E - F / G H | Pop twice , Do math and push result to stack | 4 |
|-------------------------------|--|--------------------|
| + A + / - B * C D E - F / G H | Push E | 3 4 |
| + A + / - B * C D E - F / G H | Push D | 6 3 4 |
| + A + / - B * C D E - F / G H | Push C | 2 6 3 4 |
| + A + / - B * C D E - F / G H | Pop twice, Do math and push result to stack | 12 3 4 |
| + A + / - B * C D E - F / G H | Push B | 15 12 3 4 |
| + A + / - B * C D E - F / G H | Pop twice, Do math and push result to stack | 3 3 4 |
| + A + / - B * C D E - F / G H | Pop twice, Do math and push result to stack | 1 4 |
| + A + / - B * C D E - F / G H | Pop twice, Do math and push result to stack | 5 |
| + A+/-B*CDE-F/GH | Push A | 5 |
| + A + / - B * C D E - F / G H | Pop twice, Do math and push result to stack | 25 |

ii)

Infix to postfix

A B C < C D > ||! &&! C E < ||

CONVERSION

| Expression | Action | Stack | Effect on postfix |
|---|--|-----------------------------|-------------------|
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher, Push!</td><td>!</td><td></td></e)<></c) (c> | It is an operator, its precedence is higher, Push! | ! | |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push (</td><td>!</td><td></td></e)<></c) (c> | Push (| ! | |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append A to postfix</td><td>!</td><td>A</td></e)<></c) (c> | It is an operand, Append A to postfix | ! | A |
| !(A &&!((B < C) (C > D))) (C < E) | It is an operator, its precedence is higher, Push && | <u>&&</u> (! | A |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, İts precedence is higher , Push!</td><td>! && (!</td><td>A</td></e)<></c) (c> | It is an operator, İts precedence is higher , Push! | ! && (! | A |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push (</td><td>(! && (!</td><td>A</td></e)<></c) (c> | Push (| (! && (! | A |

| 1/4001//0.5/11/6.5/11/6.5/ | Duch / | | |
|---|--|---|---------|
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push (</td><td>(! && (!</td><td>A</td></e)<></c) (c> | Push (| (! && (! | A |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append B to postfix</td><td>((! && (!</td><td>A B</td></e)<></c) (c> | It is an operand, Append B to postfix | ((! && (! | A B |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher , Push <</td><td>((! && (!</td><td>АВ</td></e)<></c) (c> | It is an operator, its precedence is higher , Push < | ((! && (! | АВ |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append C to postfix</td><td>< ((! && (! ! ! ! ! ! ! ! ! ! ! ! ! !</td><td>ABC</td></e)<></c) (c> | It is an operand, Append C to postfix | < ((! && (! ! ! ! ! ! ! ! ! ! ! ! ! ! | ABC |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Pop and append to postfix until precedence is equal or less than)</td><td>(! && (!</td><td>A B C <</td></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | (! && (! | A B C < |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, İts precedence is higher , Push </td><td> </td><td>A B C <</td></e)<></c) (c> | It is an operator, İts precedence is higher , Push | | A B C < |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push (</td><td>(</td><td>A B C <</td></e)<></c) (c> | Push (| (| A B C < |

| | | (| |
|---|--|--|-----------------------|
| | | <u> </u> | |
| | | | |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operand, Append C to postfix</th><th>(</th><th>ABC<c< th=""></c<></th></e)<></c) (c> | It is an operand, Append C to postfix | (| ABC <c< th=""></c<> |
| | | ! | |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operator, its precedence is higher , Push ></th><th>> (</th><th>A B C < C</th></e)<></c) (c> | It is an operator, its precedence is higher , Push > | > (| A B C < C |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operand, Append D to postfix</th><th>> (</th><th>ABC<cd< th=""></cd<></th></e)<></c) (c> | It is an operand, Append D to postfix | > (| ABC <cd< th=""></cd<> |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to postfix until precedence is equal or less than)</th><th> (! && (!</th><th>ABC<cd></cd></th></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | (! && (! | ABC <cd></cd> |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to postfix until precedence is equal or less than)</th><th>! && (</th><th>A B C < C D > </th></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | ! && (| A B C < C D > |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to postfix until precedence is equal or less than)</th><th>!</th><th>A B C < C D > ! &&</th></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | ! | A B C < C D > ! && |

| !(A &&!((B < C) (C > D))) (C < E) | Pop and append to postfix until precedence is equal or less than and Push | П | ABC <cd> !&&!</cd> |
|--|--|--------|-------------------------------------|
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push (</td><td>(</td><td>ABC<cd> !&&!</cd></td></e)<></c) (c> | Push (| (| ABC <cd> !&&!</cd> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append C to postfix</td><td>(!</td><td>ABC<cd> !&&!C</cd></td></e)<></c) (c> | It is an operand, Append C to postfix | (! | ABC <cd> !&&!C</cd> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher , Push <</td><td>(</td><td>ABC<cd> !&&!C</cd></td></e)<></c) (c> | It is an operator, its precedence is higher , Push < | (| ABC <cd> !&&!C</cd> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append E to postfix</td><td>< (</td><td>ABC<cd> !&&!CE</cd></td></e)<></c) (c> | It is an operand, Append E to postfix | < (| ABC <cd> !&&!CE</cd> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Pop and append to postfix until precedence is equal or less than)</td><td>П</td><td>ABC<cd> !&&! CE<</cd></td></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | П | ABC <cd> !&&! CE<</cd> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>When it is ended Pop and append to postfix until stack is empty</td><td></td><td>ABC<cd> !&&!CE< </cd></td></e)<></c) (c> | When it is ended Pop and append to postfix until stack is empty | | ABC <cd> !&&!CE< </cd> |

ii)

POSTFIX

EVALUATION

SCANNING LEFT TO RIGHT

A = 20, B = 15, C = 2, D = 6, E = 3

| Expression | Action | Stack |
|-------------------------------------|--|-------------------|
| A B C < C D > ! && ! C E < | Push A | 20 |
| A B C < C D > ! &&! C E < | Push B | 15 20 |
| A B C < C D > ! &&! C E < | Push C | 2 15 20 |
| A B C < C D > ! &&! C E < | Pop twice , Do math and push result to stack | 20 |
| A B C < C D > ! &&! C E < | Push C | 2 0 20 |
| A B C < C D > ! && ! C E < | Push D | 6 2 0 20 |
| ABC <cd> !&&!CE< </cd> | Pop twice, Do math and push result to stack | 0 0 20 |
| ABC <cd> !&&!CE< </cd> | Pop twice, Do math and push result to stack | 20 |
| ABC <cd> !&&!CE< </cd> | Pop once, Do math and push result to stack | 1 20 |
| A B C < C D > ! &&! C E < | Pop twice, Do math and push result to stack | 1 |
| ABC <cd> !&&!CE< </cd> | Pop once, Do math and push result to stack | 0 |

| ABC <cd> !&&!CE< </cd> | Push C | 0 |
|-------------------------------------|---|-------------|
| A B C < C D > ! && ! C E < | Push E | 3 2 0 |
| A B C < C D > ! && ! C E < | Pop twice, Do math and push result to stack | 0 |
| ABC <cd> !&&!CE< </cd> | Pop twice, Do math and push result to stack | 1 |

ii)

Infix to prefix

! || && A ! || < B C > C D < C E

CONVERSION

SCANNING RIGHT TO LEFT

| Expression | Action | Stack | Effect on postfix |
|---|---|----------|-------------------|
| ! (A && ! ((B < C) (C > D))) (C < E) | Push) |) | |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operand, Append E to prefix |) | Е |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operator, its precedence is higher, Push < | <u> </u> | E |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operand, Append C to prefix | <) | E C |

| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to postfix until precedence is equal or less than)</th><th></th><th>E C <</th></e)<></c) (c> | Pop and append to postfix until precedence is equal or less than) | | E C < |
|---|--|-------------------------|----------------------|
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher, Push </td><td>П</td><td>E C <</td></e)<></c) (c> | It is an operator, its precedence is higher, Push | П | E C < |
| ! (A && ! ((B < C) (C > D))) (C < E) | Push) |) | EC< |
| ! (A &&! ((B < C) (C > D))) (C < E) | Push) |) | E C < |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Push)</td><td>)))</td><td>E C <</td></e)<></c) (c> | Push) |))) | E C < |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append D to prefix</td><td>)))</td><td>E C < D</td></e)<></c) (c> | It is an operand, Append D to prefix |))) | E C < D |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher , Push ></td><td>)))]</td><td>E C < D</td></e)<></c) (c> | It is an operator, its precedence is higher , Push > |)))] | E C < D |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operand, Append C to prefix</td><td>>)))</td><td>EC<dc< td=""></dc<></td></e)<></c) (c> | It is an operand, Append C to prefix | >))) | EC <dc< td=""></dc<> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>Pop and append to prefix until precedence is equal or less than (</td><td>)</td><td>EC<dc></dc></td></e)<></c) (c> | Pop and append to prefix until precedence is equal or less than (|) | EC <dc></dc> |
| !(A&&!((B <c) (c>D))) (C<e)< td=""><td>It is an operator, its precedence is higher , Push </td><td>))</td><td>EC<dc></dc></td></e)<></c) (c> | It is an operator, its precedence is higher , Push |)) | EC <dc></dc> |
| ! (A &&! ((B < C) (C > D))) (C < E) | Push) |))) | E C < D C > |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operand, Append C to prefix |) | E C < D C > C |

| | 1 | | |
|---|--|---------------------------------|-----------------------------------|
| | |)) | |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operator, its precedence is higher , Push <</th><th><) </th><th>EC<dc>C</dc></th></e)<></c) (c> | It is an operator, its precedence is higher , Push < | <) | EC <dc>C</dc> |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operand, Append B to prefix</th><th><) </th><th>EC<dc>CB</dc></th></e)<></c) (c> | It is an operand, Append B to prefix | <) | EC <dc>CB</dc> |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to prefix until precedence is equal or less than (</th><th>)</th><th>EC < DC > CB <</th></e)<></c) (c> | Pop and append to prefix until precedence is equal or less than (|) | EC < DC > CB < |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to prefix until precedence is equal or less than (</th><th>)</th><th>E C < D C > C B < </th></e)<></c) (c> | Pop and append to prefix until precedence is equal or less than (|) | E C < D C > C B < |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>It is an operator, its precedence is higher, Push!</th><th>!) </th><th>E C < D C > C B < </th></e)<></c) (c> | It is an operator, its precedence is higher, Push! | !) | E C < D C > C B < |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operator, its precedence is lower, pop and append to prefix, then push! | &&) | E C < D C > C B < ! |
| ! (A &&! ((B < C) (C > D))) (C < E) | It is an operand, Append A to prefix | &&) | E C < D C > C B < ! A |
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>Pop and append to prefix until precedence is equal or less than (</th><th></th><th>E C < D C > C B < ! A &&</th></e)<></c) (c> | Pop and append to prefix until precedence is equal or less than (| | E C < D C > C B < ! A && |
| ! (A && ! ((B < C) (C > D))) (C < E) | It is an operator, its precedence is higher , Push! | ! | E C < D C > C B < ! A && |
| ! (A && ! ((B < C) (C > D))) (C < E) | When it is ended | | EC <dc>CB< !A&&!</dc> |

| | Pop and append to prefix until stack is empty | |
|--|--|-------------------------------------|
| !(A&&!((B <c) (c>D))) (C<e)< th=""><th>When it is ended Pop and append to prefix until stack is empty</th><th>EC<dc>CB< !A&&! </dc></th></e)<></c) (c> | When it is ended Pop and append to prefix until stack is empty | EC <dc>CB< !A&&! </dc> |

We reverse the output string to get prefix expression so;

| | ! && A ! | | < B C > C D < C E

ii)

PREFIX

EVALUATION

SCANNING RIGHT TO LEFT

| Expression | Action | Stack |
|--|--|-------------------|
| !&& A! <bc>CD<ce< td=""><td>Push E</td><td>3</td></ce<></bc> | Push E | 3 |
| !&& A! <bc>CD<ce< td=""><td>Push C</td><td>3</td></ce<></bc> | Push C | 3 |
| !&& A! <bc>CD<ce< td=""><td>Pop twice , Do math and push result to stack</td><td>1</td></ce<></bc> | Pop twice , Do math and push result to stack | 1 |
| !&& A! < B C > C D < C E | Push D | 6 1 |
| !&& A! <bc>CD<ce< td=""><td>Push C</td><td>2 6 1</td></ce<></bc> | Push C | 2 6 1 |
| !&& A! < B C > C D < C E | Pop twice , Do math and push result to stack | 0 1 |
| !&& A! <bc>CD<ce< td=""><td>Push C</td><td>2 0 1</td></ce<></bc> | Push C | 2 0 1 |
| !&& A! <bc>CD<ce< td=""><td>Push B</td><td>15 2 0 1</td></ce<></bc> | Push B | 15 2 0 1 |

| !&& A! < BC > CD < CE | Pop twice, Do math and push result to stack | 0 0 1 |
|--|---|--------------|
| ! && A! < BC > CD < CE | Pop twice, Do math and push result to stack | 0 1 |
| ! && A! < B C > C D < C E | Pop once, Do math and push result to stack | 1 1 |
| !&& A! <bc>CD<ce< td=""><td>Push A</td><td>20 1 1</td></ce<></bc> | Push A | 20 1 1 |
| ! && A! < B C > C D < C E | Pop twice, Do math and push result to stack | 1 1 |
| !&& A! <bc>CD<ce< td=""><td>Pop once, Do math and push result to stack</td><td>0 1</td></ce<></bc> | Pop once, Do math and push result to stack | 0 1 |
| !&& A! <bc>CD<ce< td=""><td>Pop once, Do math and push result to stack</td><td>1</td></ce<></bc> | Pop once, Do math and push result to stack | 1 |