Danny Tan

CS2134

HW#8

1a)

1 2 - 3 2 ^ +

input stack

1 1

2 1 2

- - 1

3 -1 3

2 -1 3 2

^ -1 9

+ 8

b)

2 3 ^ 2 ^

input stack

2 2

3 2 3

^ 8

2 8 2

^ 64

c)

2 3 2 ^ ^

input stack

2 2

3 2 3

2 2 3 2

^ 2 9

^ 512

d)

2 6 + 3 / 32 4 7 \* + 2 \* -

input stack

2 2

6 2 6

+ 8

3 8 3

/ 8/3

32 8/3 32

4 8/3 32 4

7 8/3 32 4 7

\* 8/3 32 28

+ 8/3 60

2 8/3 60 2

\* 8/3 120

- -352/3

e)

3 2 + 4 – 5 +

input stack

3 3

2 3 2

+ 5

4 5 4

- 1

5 1 5

+ 6

f)

3 2 + 4 3 2 \*4 + ^ ^

input stack

3 3

2 3 2

+ 5

4 5 4

3 5 4 3

2 5 4 3 2

\* 5 4 6

4 5 4 6 4

+ 5 4 10

^ 4 ^ 10

^ 5 ^ (4^10)

2a)

input stack

4 4

2 4 2

+ 6

3 6 3

3 6 3 3

^ 6 27

- -21

b)

input stack

3 3

2 3 2

^ 9

3 9 3

2 9 3 2

\* 9 6

- 3

c)

input stack

4 4

2 4 2

3 4 2 3

\* 4 6

- -2

3 -2 3

2 -2 3 2

^ -2 9

- -11

6 -11 6

+ -5

d)

input stack

4 4

3 4 3

+ 7

2 7 2

\* 14

1 14 1

- 13

e)

input stack

3 3

5 3 5

\* 15

1 15 1

+ 16

4 16 4

/ 4

6 4 6

+ 10

3)

enum TokenType { EOL, VALUE, OPAREN, CPAREN, EXP,

MULT, DIV, %, PLUS, MINUS };

vector PREC\_TABLE = {

{ 0, -1 }, { 0, 0 }, // EOL, VALUE

{ 100, 0 }, { 0, 99 }, // OPAREN, CPAREN

{ 6, 5 }, // EXP

{ 3, 4 }, { 3, 4 }, {3,4} // MULT, DIV, %

{ 1, 2 }, { 1, 2 } // PLUS, MINUS

};

4)

DIV. PLUS. EOL.

5a) 3

b) –

c) 4 , 5, 8 , 3

d) \* , + , -

e) 1

f) 2

g) 7

h) 4 , -

i) +

j) (4 + (5-8) )\* (3)

k) \* + 4 – 5 8 3

l) 4 5 8 - + 3 \*

6) If the height is big, then it will take more time to go through the tree. Best scenario is when all the children node is split evenly and the run time will be O(logn). The worst case is when every node has only one child and the run time will be O(n).

7) 3

1

2

3

2

1 3

3

2

1

8)

Minimal height

4

2 6

1 3 5 7

0

Maximum height

0

1

2

3

4

5

6

7

9)a)

389

39 666

30 47 400 999

9 33

b)

315

45 666

30 47 389 999

9 33 400

c)

315

39 666

30 47 389 999

9 33 45 400

398