CS 32 Week 4
Discussion 2E

Srinath

Announcements

- Homework 2 is Up!
- Due : 11:00 PM Wednesday, February 8

- Midterm : February 6, Monday evening
- Covering topics up to and including linked lists (not stacks and queues, not inheritance)

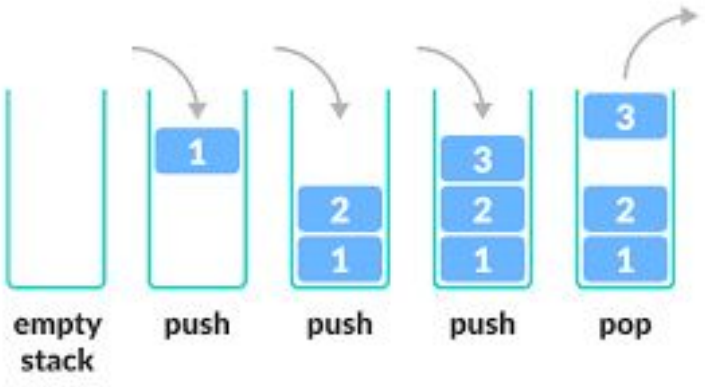
Outline

- Stack
- Queue
- Problems
- Worksheet 4

Stack

Stack : What is a Stack?

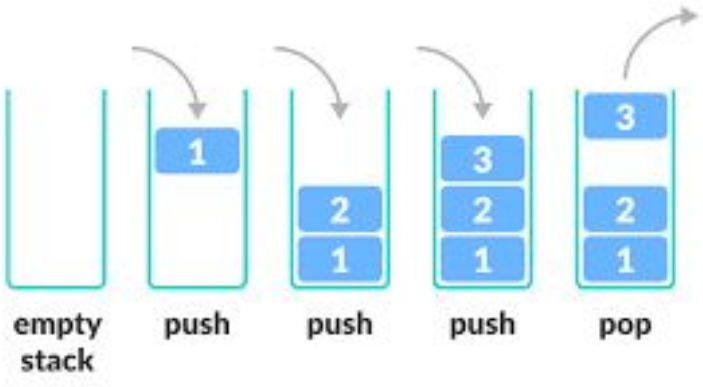
A data structure which stores some items, user of stack only has access to the last inserted element i,e Last in First Out(LIFO)



Stack : What is a Stack?

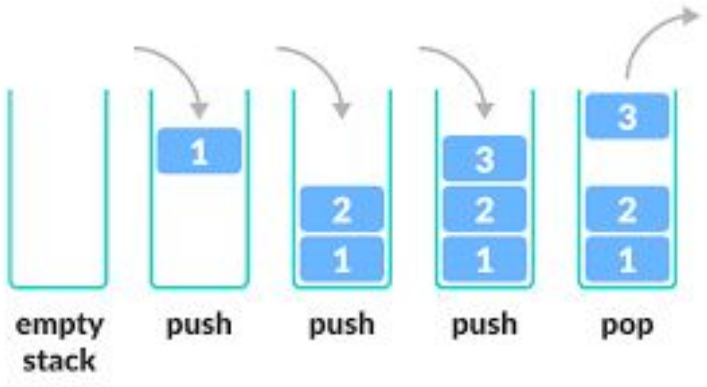
A data structure which stores some items, user of stack only has access to the last inserted element i,e Last in First Out(LIFO)

Some examples?



Stack : What is a Stack?

A data structure which stores some items, user of stack only has access to the last inserted element i,e Last in First Out(LIFO)



Some examples?

- Stack of trays in cafeteria
- Stack of plates in a cupboard
- Function call stack in a program
- Undo operation in an application

Stack : Properties

Insertion : We can only insert an element on top of a stack

Access : We can only access the top element of a stack(if stack is not empty)

Deletion : We can only remove the top element of the stack

Stack : The C++ Stack

```
#include <stack>
using namespace std;

stack<TYPENAME> s;

TYPENAME item;

// push item on top of the stack
s.push(item);

// get the top most element
ITEMTYPE topItem = s.top();  // undefined behaviour if stack is empty

// remove the top most element
s.pop();  // undefined behaviour if stack is empty

// check if the stack is empty
bool isEmpty = s.empty();

// get the size of the stack
int stack_size = s.size();
```

Stack : The C++ Stack

```
#include <stack>
using namespace std;

stack<TYPENAME> s;

TYPENAME item;

// push item on top of the stack
s.push(item);

// get the top most element
ITEMTYPE topItem = s.top();  // undefined behaviour if stack is empty

// remove the top most element
s.pop();  // undefined behaviour if stack is empty

// check if the stack is empty
bool isEmpty = s.empty();

// get the size of the stack
int stack_size = s.size();
```

- s.push(item)
- s.top()
- s.pop()
- s.empty()
- s.size()

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;
```

[ ] ←top

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;
```

[ ] ←top

```
s.push(23);
```

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;          |    [ ] ←top

s.push(23);            |    [23] ←top

s.push(45);            |
```

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;                    |    [ ] ←top

s.push(23);                      |    [23] ←top

s.push(45);                      |    [23, 45] ←top

s.push(67); s.push(98);          |
```

Stack : An Example

```
#include <stack>
using namespace std;
```

`stack<int> s;`	[ ] ←top
`s.push(23);`	[23] ←top
`s.push(45);`	[23, 45] ←top
`s.push(67); s.push(98);`	[23, 45, 67, 98] ←top
`s.top();`	

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;                 |    [ ] ←top

s.push(23);                   |    [23] ←top

s.push(45);                   |    [23, 45] ←top

s.push(67); s.push(98);       |    [23, 45, 67, 98] ←top

s.top();                      |    [23, 45, 67, 98] ←top        // returns 98

s.empty();                    |
```

Stack : An Example

```
#include <stack>
using namespace std;
```

`stack<int> s;`	[ ] ←top	
`s.push(23);`	[23] ←top	
`s.push(45);`	[23, 45] ←top	
`s.push(67); s.push(98);`	[23, 45, 67, 98] ←top	
`s.top();`	[23, 45, 67, 98] ←top	// returns 98
`s.empty();`	[23, 45, 67, 98] ←top	// returns False
`s.pop();`		

Stack : An Example

```
#include <stack>
using namespace std;
```

stack<int> s;	[ ] ←top	
s.push(23);	[23] ←top	
s.push(45);	[23, 45] ←top	
s.push(67); s.push(98);	[23, 45, 67, 98] ←top	
s.top();	[23, 45, 67, 98] ←top	// returns 98
s.empty();	[23, 45, 67, 98] ←top	// returns False
s.pop();	[23, 45, 67] ←top	// no return value
s.size();		

Stack : An Example

```
#include <stack>
using namespace std;

stack<int> s;                 |   [ ] ←top
s.push(23);                   |   [23] ←top
s.push(45);                   |   [23, 45] ←top
s.push(67); s.push(98);       |   [23, 45, 67, 98] ←top
s.top();                      |   [23, 45, 67, 98] ←top      // returns 98
s.empty();                    |   [23, 45, 67, 98] ←top      // returns False
s.pop();                      |   [23, 45, 67] ←top          // no return value
s.size();                     |   [23, 45, 67] ←top          // returns 3
s.top();                      |
```

Stack : An Example

```
#include <stack>
using namespace std;
```

stack<int> s;	[ ] ←top	
s.push(23);	[23] ←top	
s.push(45);	[23, 45] ←top	
s.push(67); s.push(98);	[23, 45, 67, 98] ←top	
s.top();	[23, 45, 67, 98] ←top	// returns 98
s.empty();	[23, 45, 67, 98] ←top	// returns False
s.pop();	[23, 45, 67] ←top	// no return value
s.size();	[23, 45, 67] ←top	// returns 3
s.top();	[23, 45, 67] ←top	// returns 67

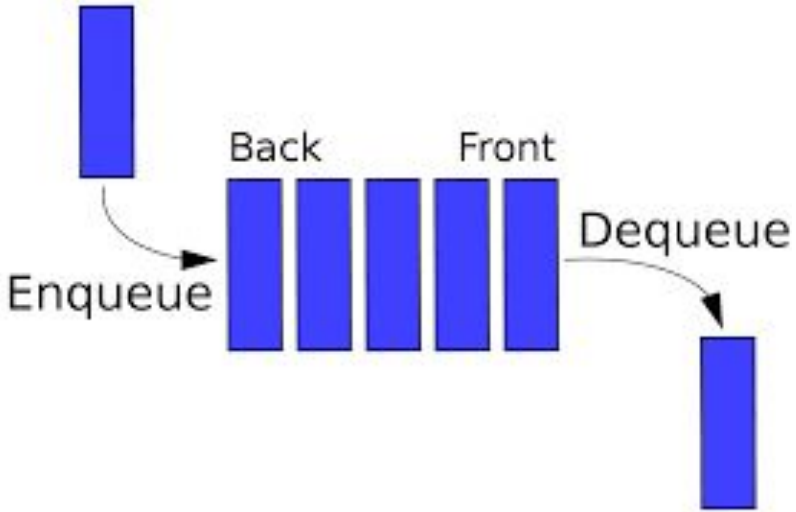
Stack : Implementation - Array

Stack : Implementation - Linked List

Queue

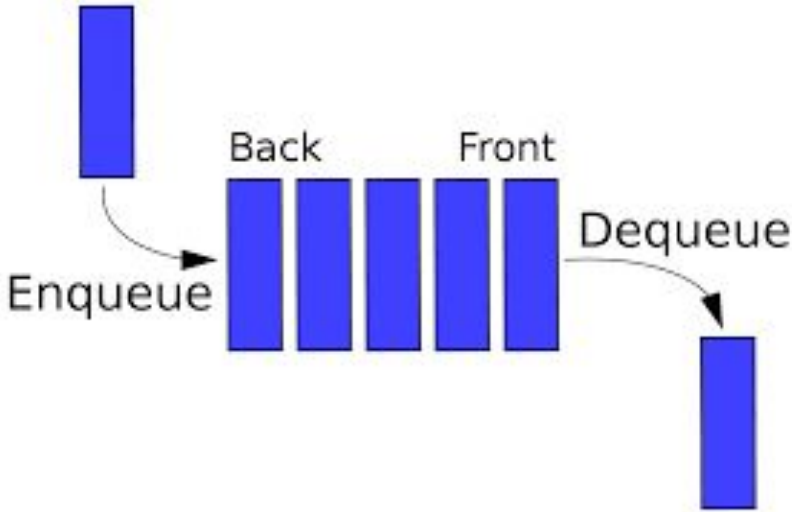
Queue : What is a Queue?

A data structure which stores some items, user of queue inserts elements from one end while accesses them from the other end.
First In First Out(FIFO)



Queue : What is a Queue?

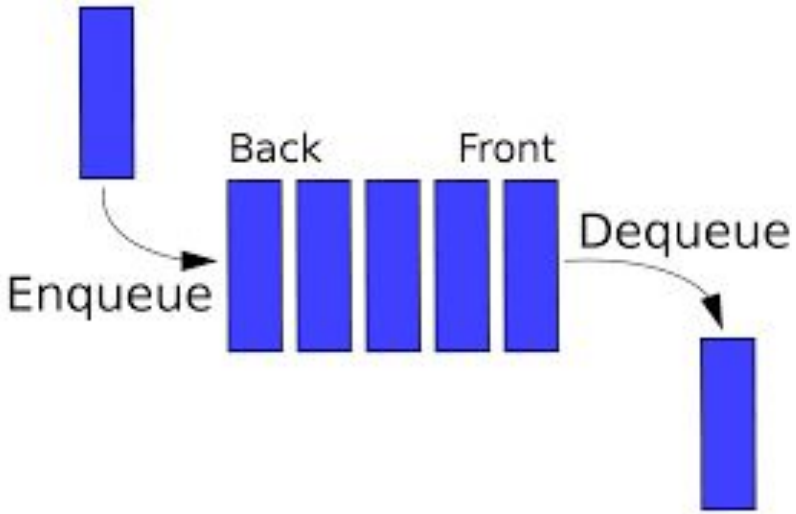
A data structure which stores some items, user of queue inserts elements from one end while accesses them from the other end.
First In First Out(FIFO)



Some examples?

Queue : What is a Queue?

A data structure which stores some items, user of queue inserts elements from one end while accesses them from the other end.
First In First Out(FIFO)



Some examples?

- Queue for a movie ticket
- A car wash
- A printer queue

Queue : Properties

Insertion : We can only insert an element at back of a queue

Access : We can only access the element at front of a queue (there are variations which also gives access to element at back of the queue)

Deletion : We can only remove the element at front of a queue

Queue : The C++ Queue

```
#include <queue>
using namespace std;

queue<TYPENAME> q;
TYPENAME item;

// push item into the queue (enqueue)
q.push(item);

// get the front element of the queue
ITEMTYPE frontItem = q.front();  // undefined behaviour if queue is empty

// get the back element of the queue
ITEMTYPE backItem = q.back();  // undefined behaviour if queue is empty

// remove the front element of the queue (dequeue)
q.pop();  // undefined behaviour if queue is empty

// check if the queue is empty
bool isEmpty = q.empty();

// get the size of the queue
int queue_size = q.size();
```

Queue : The C++ Queue

```
#include <queue>
using namespace std;

queue<TYPENAME> q;
TYPENAME item;

// push item into the queue (enqueue)
q.push(item);

// get the front element of the queue
ITEMTYPE frontItem = q.front();  // undefined behaviour if queue is empty

// get the back element of the queue
ITEMTYPE backItem = q.back();  // undefined behaviour if queue is empty

// remove the front element of the queue (dequeue)
q.pop();  // undefined behaviour if queue is empty

// check if the queue is empty
bool isEmpty = q.empty();

// get the size of the queue
int queue_size = q.size();
```

- q.push(item)
- q.front()
- q.back()
- q.pop()
- q.empty()
- q.size()

Queue : An Example

```
#include <queue>
using namespace std;

queue<int> q;
```

back→[ ] ←front

```
q.push(23);
```

Queue : An Example

```
#include <queue>
using namespace std;

queue<int> q;          |    back→[ ] ←front

q.push(23);            |    back→[23] ←front

q.push(45);            |
```

Queue : An Example

```
#include <queue>
using namespace std;
```

`queue<int> q;`	back→[ ] ←front
`q.push(23);`	back→[23] ←front
`q.push(45);`	back→[45, 23] ←front
`q.push(67); q.push(98);`	

Queue : An Example

```
#include <queue>
using namespace std;
```

`queue<int> q;`	back→[ ] ←front
`q.push(23);`	back→[23] ←front
`q.push(45);`	back→[45, 23] ←front
`q.push(67); q.push(98);`	back→[98, 67, 45, 23] ←front
`q.front();`	

Queue : An Example

```
#include <queue>
using namespace std;
```

queue<int> q;	back→[ ] ←front	
q.push(23);	back→[23] ←front	
q.push(45);	back→[45, 23] ←front	
q.push(67); q.push(98);	back→[98, 67, 45, 23] ←front	
q.front();	back→[98, 67, 45, 23] ←front	// returns 23
q.empty();		

Queue : An Example

```
#include <queue>
using namespace std;
```

queue<int> q;	back→[ ] ←front	
q.push(23);	back→[23] ←front	
q.push(45);	back→[45, 23] ←front	
q.push(67); q.push(98);	back→[98, 67, 45, 23] ←front	
q.front();	back→[98, 67, 45, 23] ←front	// returns 23
q.empty();	back→[98, 67, 45, 23] ←front	// returns False
q.pop();		

Queue : An Example

```
#include <queue>
using namespace std;
```

queue<int> q;	back→[ ] ←front	
q.push(23);	back→[23] ←front	
q.push(45);	back→[45, 23] ←front	
q.push(67); q.push(98);	back→[98, 67, 45, 23] ←front	
q.front();	back→[98, 67, 45, 23] ←front	// returns 23
q.empty();	back→[98, 67, 45, 23] ←front	// returns False
q.pop();	back→[98, 67, 45] ←front	// no return value
q.size();		

Queue : An Example

```
#include <queue>
using namespace std;
```

queue<int> q;	back→[ ] ←front	
q.push(23);	back→[23] ←front	
q.push(45);	back→[45, 23] ←front	
q.push(67); q.push(98);	back→[98, 67, 45, 23] ←front	
q.front();	back→[98, 67, 45, 23] ←front	// returns 23
q.empty();	back→[98, 67, 45, 23] ←front	// returns False
q.pop();	back→[98, 67, 45] ←front	// no return value
q.size();	back→[98, 67, 45] ←front	// returns 3
q.back();		

Queue : An Example

```
#include <queue>
using namespace std;
```

queue<int> q;	back→[ ] ←front	
q.push(23);	back→[23] ←front	
q.push(45);	back→[45, 23] ←front	
q.push(67); q.push(98);	back→[98, 67, 45, 23] ←front	
q.front();	back→[98, 67, 45, 23] ←front	// returns 23
q.empty();	back→[98, 67, 45, 23] ←front	// returns False
q.pop();	back→[98, 67, 45] ←front	// no return value
q.size();	back→[98, 67, 45] ←front	// returns 3
q.back();	back→[98, 67, 45] ←front	// returns 98

Queue : Implementation - Array

Queue : Implementation - Linked List

Problems

Problems : Background

Prefix, Infix and Postfix expressions

Problems : Background

Prefix, Infix and Postfix expressions

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 |

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Postfix : operand1 operand2 operator |

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2	14 / 7
Prefix : operator operand1 operand2	/ 14 7
Postfix : operand1 operand2 operator	14 7 /

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Postfix : operand1 operand2 operator | 14 7 /

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Infix : 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)

Prefix :

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Postfix : operand1 operand2 operator | 14 7 /

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Infix : 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)

Prefix : + 3 - 8 + 1 - / 14 7 6

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Postfix : operand1 operand2 operator | 14 7 /

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Infix : 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)

Prefix : + 3 - 8 + 1 - / 14 7 6

Postfix :

Problems : Background

Prefix, Infix and Postfix expressions

Infix : operand1 operator operand2 | 14 / 7

Prefix : operator operand1 operand2 | / 14 7

Postfix : operand1 operand2 operator | 14 7 /

Operator : +, - , / , !, ||, && etc..

Operand : elements on which we apply the operator
eg : numbers, booleans etc.

Infix : 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)

Prefix : + 3 - 8 + 1 - / 14 7 6

Postfix : 3 8 + 1 - 14 7 / + 6 -

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

Stack : []

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

Stack : []
Stack : [3]

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]
We see +, so compute 10 + 2, push
Stack : [12]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]
We see +, so compute 10 + 2, push
Stack : [12]
Stack : [12 6]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]
We see +, so compute 10 + 2, push
Stack : [12]
Stack : [12 6]
We see -, so compute 12 - 6, push
Stack : [6]

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]
We see +, so compute 10 + 2, push
Stack : [12]
Stack : [12 6]
We see -, so compute 12 - 6, push
Stack : [6]
End of expression
Result = 6

Problems : Postfix evaluation

Given a postfix expression, 3 8 + 1 - 14 7 / + 6 -
How to evaluate it? (we know all operators are binary)

We use a Stack to do this

Start with an empty stack

If operand? Push it on top

If operator? Remove top 2 elements, compute and push the result on top

Repeat until you finish the whole expression

Your stack will be left with 1 element at the end, which is the result

What if we have unary operators like '!' ??

Stack : []
Stack : [3]
Stack : [3 8]
We see +, so compute 3 + 8, push
Stack : [11]
Stack : [11 1]
We see -, so compute 11 - 1, push
Stack : [10]
Stack : [10 14]
Stack : [10 14 7]
We see /, so compute 14 / 7, push
Stack : [10 2]
We see +, so compute 10 + 2, push
Stack : [12]
Stack : [12 6]
We see -, so compute 12 - 6, push
Stack : [6]
End of expression
Result = 6

Problems : Infix to Postfix

Given an infix expression, `3 + 8 - 1 + 14 / 7 - 6`, operator precedence : /, (+, -)
How to convert it to postfix?

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

Stack : [] result : “”

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

Stack : [] result : “”
Stack : [] result : “3”

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [] result : “”
Stack : [] result : “3”
Stack : [+] result : “3”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : []	result : “”
Stack : []	result : “3”
Stack : [+]	result : “3”
Stack : [+]	result : “3 8”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [] result : “”
Stack : [] result : “3”
Stack : [+] result : “3”
Stack : [+] result : “3 8”
We see -
+ doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [-] result : “3 8 +”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [] result : “”
Stack : [] result : “3”
Stack : [+] result : “3”
Stack : [+] result : “3 8”

We see -
+ doesn’t have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : “3 8 +”
Stack : [-] result : “3 8 + 1”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [] result : “”
Stack : [] result : “3”
Stack : [+] result : “3”
Stack : [+] result : “3 8”
We see -
+ doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [-] result : “3 8 +”
Stack : [-] result : “3 8 + 1”
We see +
- doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [+] result : “3 8 + 1 -”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”
We see /
+ have a lower precedence
Nothing to do.
Push current operator
Stack : [+ /] result : “3 8 + 1 - 14”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”
We see /
+ have a lower precedence
Nothing to do.
Push current operator
Stack : [+ /] result : “3 8 + 1 - 14”
Stack : [+ /] result : “3 8 + 1 - 14 7”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”
We see /
+ have a lower precedence
Nothing to do.
Push current operator
Stack : [+ /] result : “3 8 + 1 - 14”
Stack : [+ /] result : “3 8 + 1 - 14 7”
We see -
/ doesn’t have lower precedence
So, pop and add to result.
+ also doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [-] result : “3 8 + 1 - 14 7 / +”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”
We see /
+ have a lower precedence
Nothing to do.
Push current operator
Stack : [+ /] result : “3 8 + 1 - 14”
Stack : [+ /] result : “3 8 + 1 - 14 7”
We see -
/ doesn’t have lower precedence
So, pop and add to result.
+ also doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [-] result : “3 8 + 1 - 14 7 / +”
Stack : [-] result : “3 8 + 1 - 14 7 / + 6”

Problems : Infix to Postfix

Given an infix expression, 3 + 8 - 1 + 14 / 7 - 6, operator precedence : /, (+, -)
How to convert it to postfix?

We also use a Stack to do this, but store operators now.
Along with stack, maintain a result string

Start with an empty stack, empty result

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

……..
Stack : [+] result : “3 8 + 1 -”
Stack : [+] result : “3 8 + 1 - 14”
We see /
+ have a lower precedence
Nothing to do.
Push current operator
Stack : [+ /] result : “3 8 + 1 - 14”
Stack : [+ /] result : “3 8 + 1 - 14 7”
We see -
/ doesn’t have lower precedence
So, pop and add to result.
+ also doesn’t have lower precedence
So, pop and add to result.
Push current operator
Stack : [-] result : “3 8 + 1 - 14 7 / +”
Stack : [-] result : “3 8 + 1 - 14 7 / + 6”
We hit END, pop everything and add to result
Stack : [] result : “3 8 + 1 - 14 7 / + 6 -”

Problems : Infix to Postfix

Did we miss anything?

Given an infix expression, ….. , operator precedence : /, (+, -)
How to convert it to postfix?

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
    Until you find an operator with
    lower precedence on stack or you hit '(' or stack is empty,
    Pop stack and add popped operator to result

    Push current operator to stack

If open parenthesis '(' ?
    Push it on top of the stack

If close parenthesis ')' ?
    Until you find the corresponding
    open parenthesis, Pop stack and
    add popped operator to result.
    Also pop '('

If end of expression?
    Until stack is empty,
    Pop stack and add popped operator to result

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

Stack : [ ] result : “”

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit ‘(‘ or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis ‘(’ ?
Push it on top of the stack

If close parenthesis ‘)’ ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop ‘(’

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [ ] result : ""
Stack : [ ] result : "3"
Stack : [+] result : "3"
Stack : [+] result : "3 8"

We see -
+ doesn't have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : "3 8 +"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [ ] result : ""
Stack : [ ] result : "3"
Stack : [+] result : "3"
Stack : [+] result : "3 8"

We see -
+ doesn't have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : "3 8 +"

We see (, push it

Stack : [- ( ] result : "3 8 +"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [ ] result : ""
Stack : [ ] result : "3"
Stack : [+] result : "3"
Stack : [+] result : "3 8"

We see -
+ doesn't have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : "3 8 +"

We see (, push it

Stack : [- ( ] result : "3 8 +"
Stack : [- ( ] result : "3 8 + 1"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [ ] result : ""
Stack : [ ] result : "3"
Stack : [+] result : "3"
Stack : [+] result : "3 8"

We see -
+ doesn't have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : "3 8 +"

We see (, push it

Stack : [- ( ] result : "3 8 +"
Stack : [- ( ] result : "3 8 + 1"

We see +,
We hit '(', Nothing to do.
Push current operator

Stack : [- (+] result : "3 8 + 1"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Stack : [ ] result : ""
Stack : [ ] result : "3"
Stack : [+] result : "3"
Stack : [+] result : "3 8"

We see -
+ doesn't have lower precedence
So, pop and add to result.
Push current operator

Stack : [-] result : "3 8 +"

We see (, push it

Stack : [- ( ] result : "3 8 +"
Stack : [- ( ] result : "3 8 + 1"

We see +,
We hit '(', Nothing to do.
Push current operator

Stack : [- (+] result : "3 8 + 1"
Stack : [- (+] result : "3 8 + 1 14"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

.....
Stack : [- (+] result : “3 8 + 1 14”

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit ‘(‘ or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis ‘(’ ?
Push it on top of the stack

If close parenthesis ‘)’ ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop ‘(’

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"
Stack : [- (+ /] result : "3 8 + 1 14 7"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"
Stack : [- (+ /] result : "3 8 + 1 14 7"
We see),
Pop and add everything until '(',
Pop '('
Stack : [-] result : "3 8 + 1 14 7 / +"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"
Stack : [- (+ /] result : "3 8 + 1 14 7"
We see),
Pop and add everything until '(',
Pop '('
Stack : [-] result : "3 8 + 1 14 7 / +"
We see -,
- doesn't have lower precedence
Pop, add to result.
Push current operator
Stack : [-] result : "3 8 + 1 14 7 / + -"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"
Stack : [- (+ /] result : "3 8 + 1 14 7"
We see),
Pop and add everything until '(',
Pop '('
Stack : [-] result : "3 8 + 1 14 7 / +"
We see -,
- doesn't have lower precedence
Pop, add to result.
Push current operator
Stack : [-] result : "3 8 + 1 14 7 / + -"
Stack : [-] result : "3 8 + 1 14 7 / + - 6"

Problems : Infix to Postfix

Handling parenthesis

Given an infix expression, 3 + 8 - (1 + 14 / 7) - 6, operator precedence : /, (+, -)
How to convert it to postfix?

If operand? Add it to the result

If operator?
Until you find an operator with
lower precedence on stack or you hit '(' or stack is empty,
Pop stack and add popped operator to result

Push current operator to stack

If open parenthesis '(' ?
Push it on top of the stack

If close parenthesis ')' ?
Until you find the corresponding
open parenthesis, Pop stack and
add popped operator to result.
Also pop '('

If end of expression?
Until stack is empty,
Pop stack and add popped operator to result

.....
Stack : [- (+] result : "3 8 + 1 14"
We see /,
+ has lower precedence,
Nothing to do.
Push current operator
Stack : [- (+ /] result : "3 8 + 1 14"
Stack : [- (+ /] result : "3 8 + 1 14 7"
We see),
Pop and add everything until '(',
Pop '('
Stack : [-] result : "3 8 + 1 14 7 / +"
We see -,
- doesn't have lower precedence
Pop, add to result.
Push current operator
Stack : [-] result : "3 8 + 1 14 7 / + -"
Stack : [-] result : "3 8 + 1 14 7 / + - 6"
We hit END, pop everything and add to result
Stack : [ ] result : "3 8 + 1 14 7 / + - 6 -"

Problems : Rolling Sum

Given K and an incoming stream of integers from stream, calculate the sum of latest K elements.

Example :
K = 3
Stream : 2, 4, 6, 8, 10, 4, 2, 60, 40, 50,.....

Output : -, -, 12, 18, 24, 22, 16, 66, 102, 150,....

Problems : Rolling Sum

Given K and an incoming stream of integers from stream, calculate the sum of latest K elements.

Example :
K = 3
Stream : 2, 4, 6, 8, 10, 4, 2, 60, 40, 50,.....

Output : -, -, 12, 18, 24, 22, 16, 66, 102, 150,....

Hint : Use a queue!!

Problems : Rolling Sum

Given K and an incoming stream of integers from stream, calculate the sum of latest K elements.

Example :
K = 3
Stream : 2, 4, 6, 8, 10, 4, 2, 60, 40, 50,.....

Output : -, -, 12, 18, 24, 22, 16, 66, 102, 150,....

Hint : Use a queue!!

Maintain a queue of K elements, variable Sum to store the sum
When you get a new element
- Subtract front element of queue from the Sum
- Add the new element to the Sum
- Pop the queue
- Add the new element to the Queue.