```
void flipStack2 (StackOfT& s)
//! updates s
//! requires |s| = 2
//! ensures s = #s[1,2) * #s[0,1)
```

Name: _____

Name: _____

One CM: _____

Reasoning Table for *flipStack2*

S	Code	Assume	Confirm
0		`|s0| = 2`	`true`
	`T y;` `StackOfT t;`	/ / / / / / / / / / /	/ / / / / /

1		`T.Init(y1) ^`
		`t1 = <> ^`
		`s1 = s0`
			`s1 /= <>`

	`s.pop(y)`	/ / / / / / / / / / /	/ / / / / /

2		`<y2> is prefix of s1 ^`
		`s2 = s1[1, |s1|) ^`
		`t2 = t1`
			`true`

	`t.push(y);`	/ / / / / / / / / / /	/ / / / / /

3		`t3 = <y2> * t2 ^`
		`s3 = s2 ^`
		`T.Init(y3)`
			`s3 /= <>`

	`s.pop(y)`	/ / / / / / / / / / /	/ / / / / /

4		`<y4> is prefix of s3 ^`
		`s4 = s3[1, |s3|) ^`
		`t4 = t3`
			`true`

	`t.push(y)`	/ / / / / / / / / / /	/ / / / / /

5		`t5 = <y4> * t4 ^`
		`s5 = s4 ^`
		`T.Init(y5)`
			`true`

	`s.transferFrom(t);`	/ / / / / / / / / / /	/ / / / / /

6		`s6 = t5 ^`
		`t6 = <> ^`
		`y6 = y5`
			`s6 = s0[1,2) * s0[0,1)`

VC1: `|s0| = 2 ^ T.Init(y1) ^ t1 = <> ^ s1 = s0 → s1 /= <>`

VC3:
```
|s0| = 2 ^ T.Init(y1) ^ t1 = <>  ^ s1 = s0 ^
<y2> is prefix of s1 ^ s2 = s1[1, |s1|) ^ t2 = t1 ^
t3 = <y2> * t2 ^ s3 = s2 ^ T.Init(y3) → s3 /= <>
```

VC6:
```
|s0| = 2 ^ T.Init(y1) ^ t1 = <>  ^ s1 = s0 ^
<y2> is prefix of s1 ^ s2 = s1[1, |s1|) ^ t2 = t1 ^
t3 = <y2> * t2 ^ s3 = s2 ^ T.Init(y3) ^
<y4> is prefix of s3 ^ s4 = s3[1, |s3|) ^ t4 = t3 ^
t5 = <y4> * t4 ^ s5 = s4 ^ T.Init(y5) ^
s6 = t5 ^ t6 = <> ^ y6 = y5 → s6 = s0[1,2) * s0[0,1)
```

Steps:

1. Assume all the *premises* on the lhs of → are true
2. Use the premises from the lhs to show that the *conclusion* on the rhs cannot be false

- The proof goes through if the conclusion is shown to be true
- The proof fails if the conclusion cannot be shown to be true

To do: Prove VC6

```
void flipStack2-Defective (StackOfT& s)
//! updates s
//! requires |s| = 2
//! ensures s = #s[1,2) * #s[0,1)
```

Name: _____

Name: _____

One CM: _____

Reasoning Table for *flipStack2-Defective*

S	Code	Assume	Confirm
0		|s0| = 2	true
	T y; StackOfT t;	/ / / / / / / / / / /	/ / / / / /
1		T.Init(y1) ^ t1 = <> ^ s1 = s0	s1 /= <>
	s.pop(y)	/ / / / / / / / / / /	/ / / / / /
2		<y2> is prefix of s1 ^ s2 = s1[1, |s1|) ^ t2 = t1	true
	t.push(y);	/ / / / / / / / / / /	/ / / / / /
3		t3 = <y2> * t2 ^ s3 = s2 ^ T.Init(y3)	s3 /= <>
	s.pop(y)	/ / / / / / / / / / /	/ / / / / /
4		<y4> is prefix of s3 ^ s4 = s3[1, |s3|) ^ t4 = t3	true
	s.push(y)	/ / / / / / / / / / /	/ / / / / /
5		s5 = <y4> * s4 ^ t5 = t4 ^ T.Init(y5)	true
	s.transferFrom(t);	/ / / / / / / / / / /	/ / / / / /
6		s6 = t5 ^ t6 = <> ^ y6 = y5	s6 = s0[1,2) * s0[0,1)

VC1: |s0| = 2 ^ T.Init(y1) ^ t1 = <> ^ s1 = s0 → s1 /= <>

VC3: |s0| = 2 ^ T.Init(y1) ^ t1 = <> ^ s1 = s0 ^
<y2> is prefix of s1 ^ s2 = s1[1, |s1|) ^ t2 = t1 ^
t3 = <y2> * t2 ^ s3 = s2 ^ T.Init(y3) → s3 /= <>

VC6: |s0| = 2 ^ T.Init(y1) ^ t1 = <> ^ s1 = s0 ^
<y2> is prefix of s1 ^ s2 = s1[1, |s1|) ^ t2 = t1 ^
t3 = <y2> * t2 ^ s3 = s2 ^ T.Init(y3) ^
<y4> is prefix of s3 ^ s4 = s3[1, |s3|) ^ t4 = t3 ^
s5 = <y4> * s4 ^ t5 = t4 ^ T.Init(y5) ^
s6 = t5 ^ t6 = <> ^ y6 = y5 → s6 = s0[1,2) * s0[0,1)

To do: Prove VC6