Writing Reusable Code Feedback at Scale with Mixed-Initiative Program Synthesis

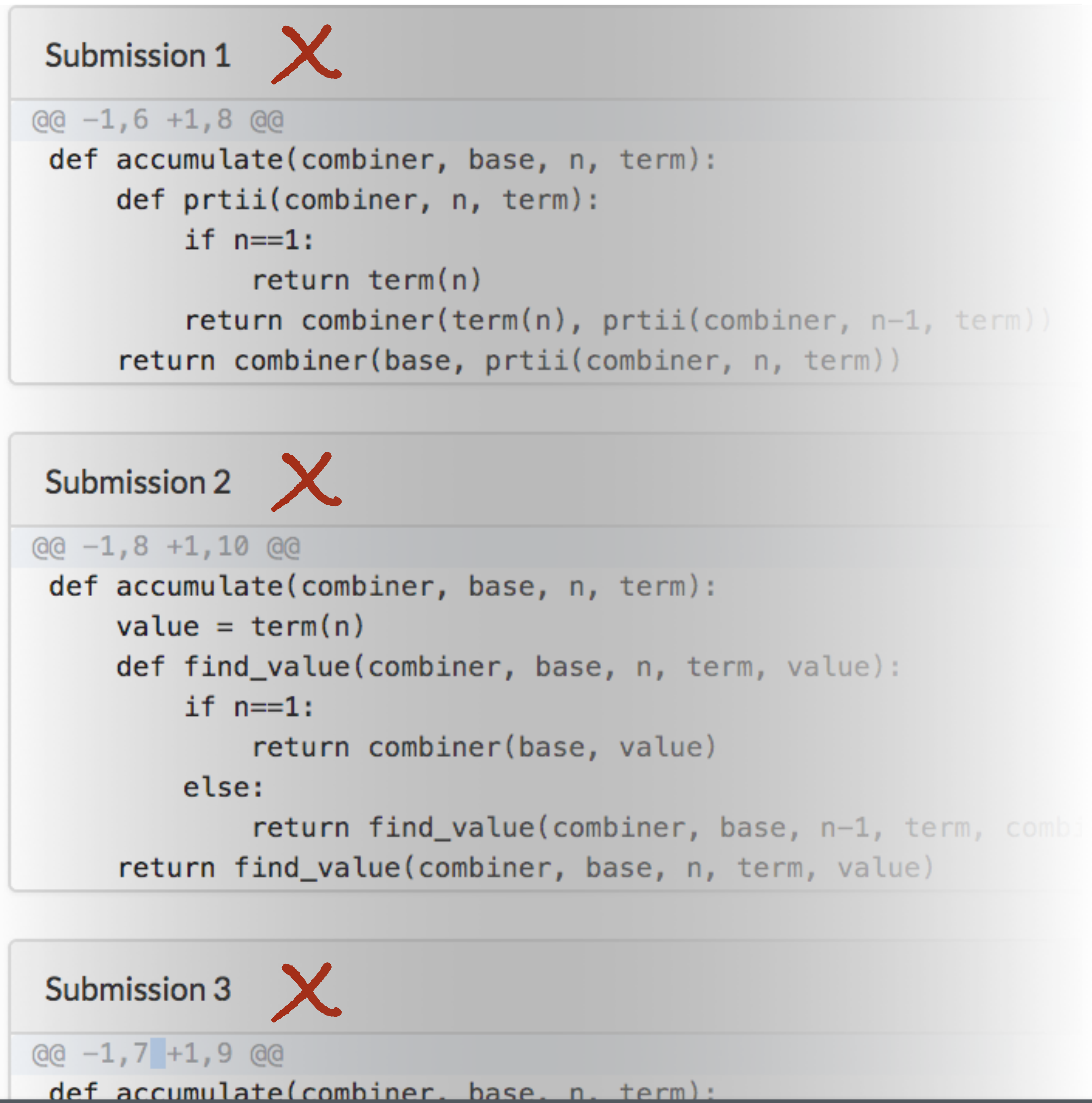
Andrew Head*, Elena Glassman*, Gustavo Soares*,
Ryo Suzuki, Lucas Figueredo,
Loris D'Antoni, Björn Hartmann

* These three authors contributed equally to the work.

Berkeley UNIVERSITY OF CALIFORNIA
Universidade Federal de Campina Grande
University of Colorado Boulder
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

When Writing Feedback on Student Code, Teachers Can Draw on Deep Domain Knowledge

Incorrect Student Code Submissions



Submission 1

```
@@ -1,6 +1,8 @@
def accumulate(combiner, base, n, term):
    def prtii(combiner, n, term):
        if n==1:
            return term(n)
        return combiner(term(n), prtii(combiner, n-1, term))
    return combiner(base, prtii(combiner, n, term))
```

Submission 2

```
@@ -1,8 +1,10 @@
def accumulate(combiner, base, n, term):
    value = term(n)
    def find_value(combiner, base, n, term, value):
        if n==1:
            return combiner(base, value)
        else:
            return find_value(combiner, base, n-1, term, comb
    return find_value(combiner, base, n, term, value)
```

Submission 3

```
@@ -1,7 +1,9 @@
def accumulate(combiner, base, n, term):
```

Teacher Comments

What happens when n is zero? Hint: look at lecture 5's slide

While this he

…but it does not scale.

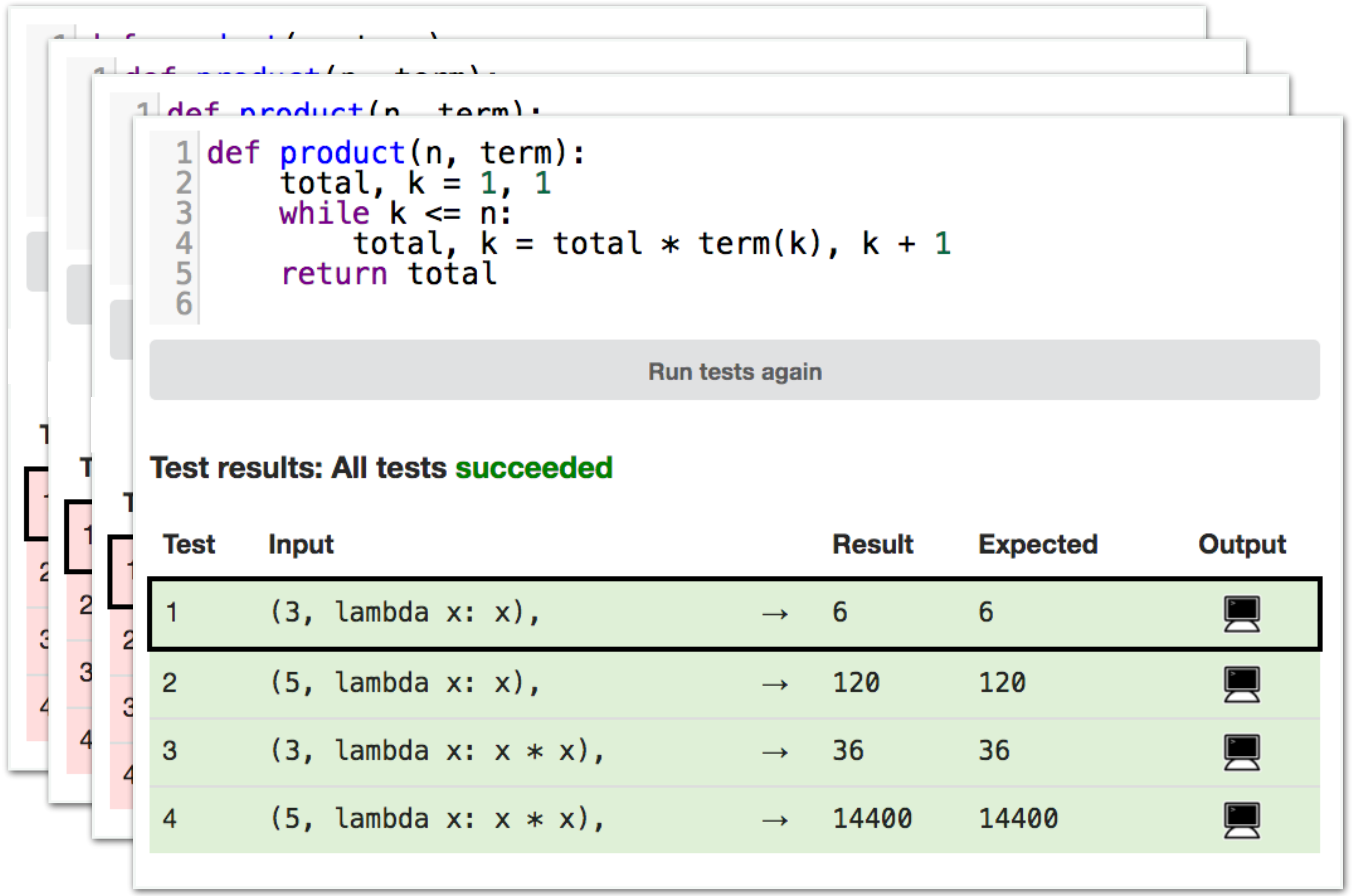
e c

Have you considered what would happen if combiner was se

In lieu of Teacher-Written Feedback, Autograder Shows Test Cases

Student Submission

Test Case Results



```
def product(n, term):
    total, k = 1, 1
    while k <= n:
        total, k = total * term(k), k + 1
    return total

```

Run tests again

Test results: All tests succeeded

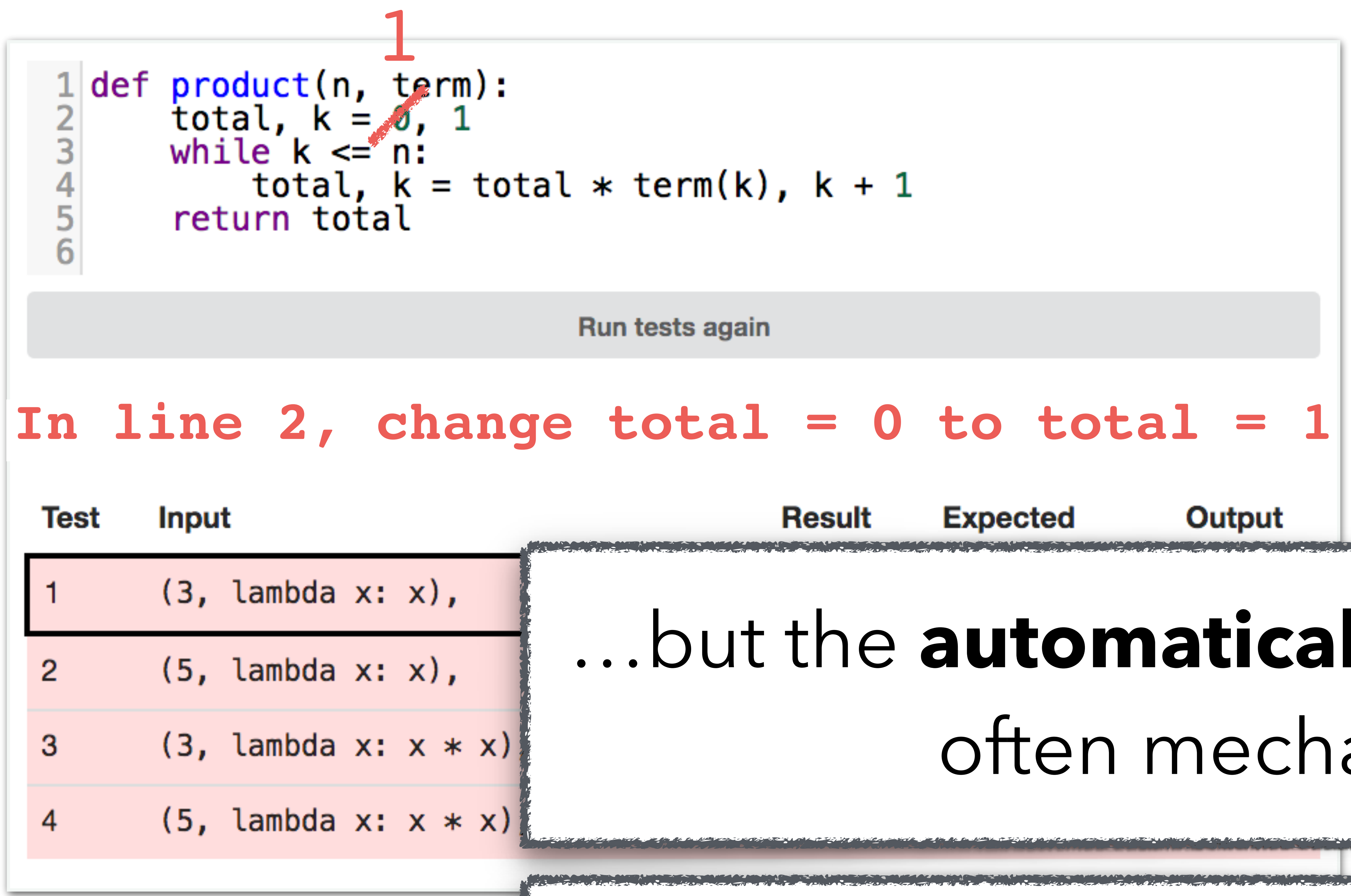
Test	Input		Result	Expected	Output
1	(3, lambda x: x),	→	6	6	💻
2	(5, lambda x: x),	→	120	120	💻
3	(3, lambda x: x * x),	→	36	36	💻
4	(5, lambda x: x * x),	→	14400	14400	💻

Course Autograder

…but there's still a **gulf of evaluation**.

Program Synthesis Techniques Can Shrink the Gulf by Automatically Finding and Suggesting Bug Fixes for Students

Student Submission



```
1 def product(n, term):
2     total, k = ~~0~~ 1, 1
3     while k <= n:
4         total, k = total * term(k), k + 1
5     return total
6
```

Run tests again

In line 2, change total = 0 to total = 1

Test Case Results

Test	Input	Result	Expected	Output
1	(3, lambda x: x),			
2	(5, lambda x: x),			
3	(3, lambda x: x * x)			
4	(5, lambda x: x * x)			

…but the **automatically generated feedback** is often mechanical, formulaic

Can we combine teachers' deep domain knowledge with program synthesis to **give students better feedback**?

Learning Code Transformations from Pairs of Incorrect and Correct Submissions

Student 1 fixes iterative solution

```
 def product(n, term):
   total, k = 1, 1
   while k<=n:
-    total = total*k
+    total = total*term(k)
     k = k+1
   return total
```

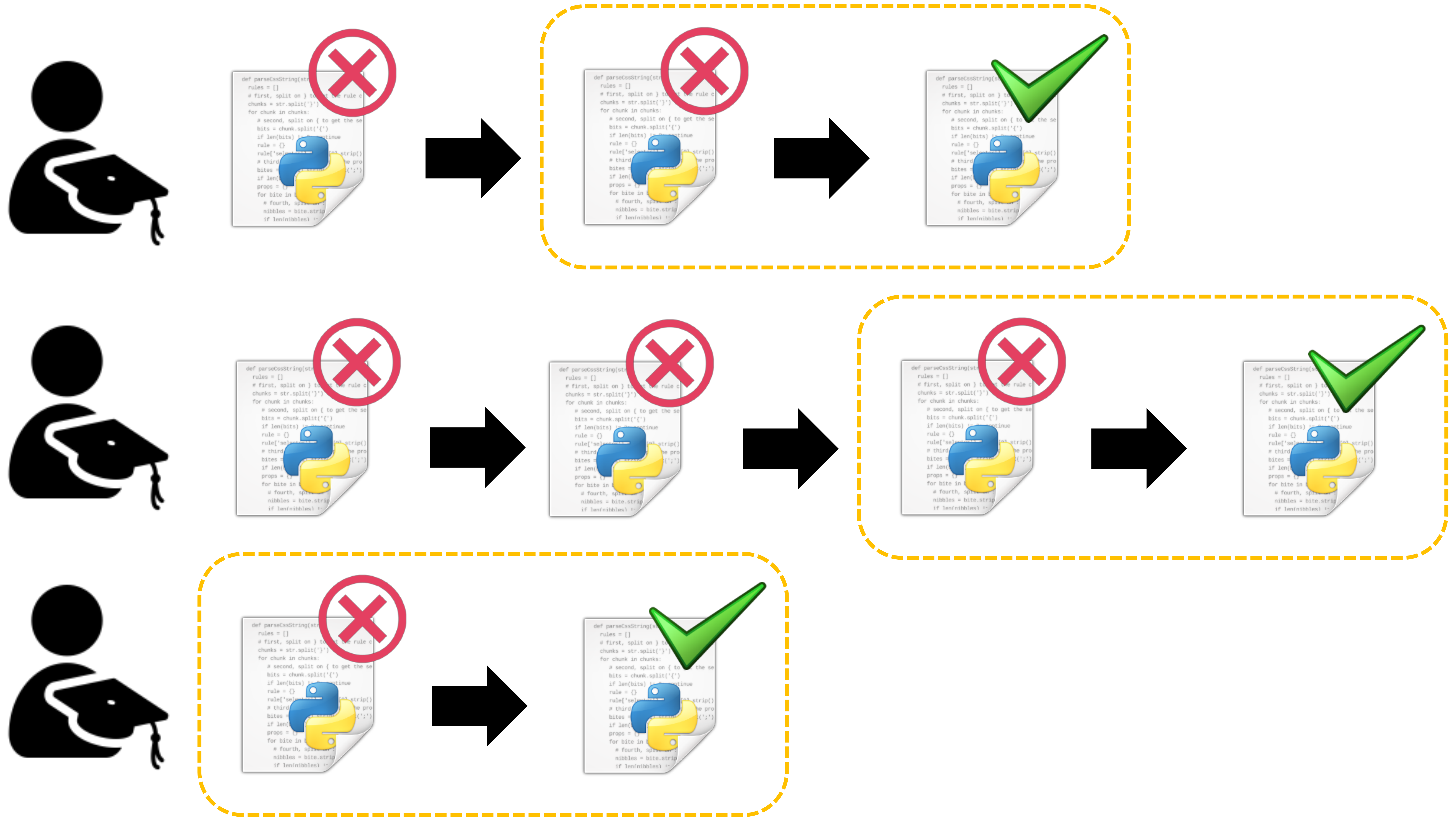
Student 2 fixes recursive solution

```
 def product(n, term):
   if (n==1):
     return 1
-  return product(n-1, term)*n
+  return product(n-1, term)*term(n)
```

Generalized code transformation

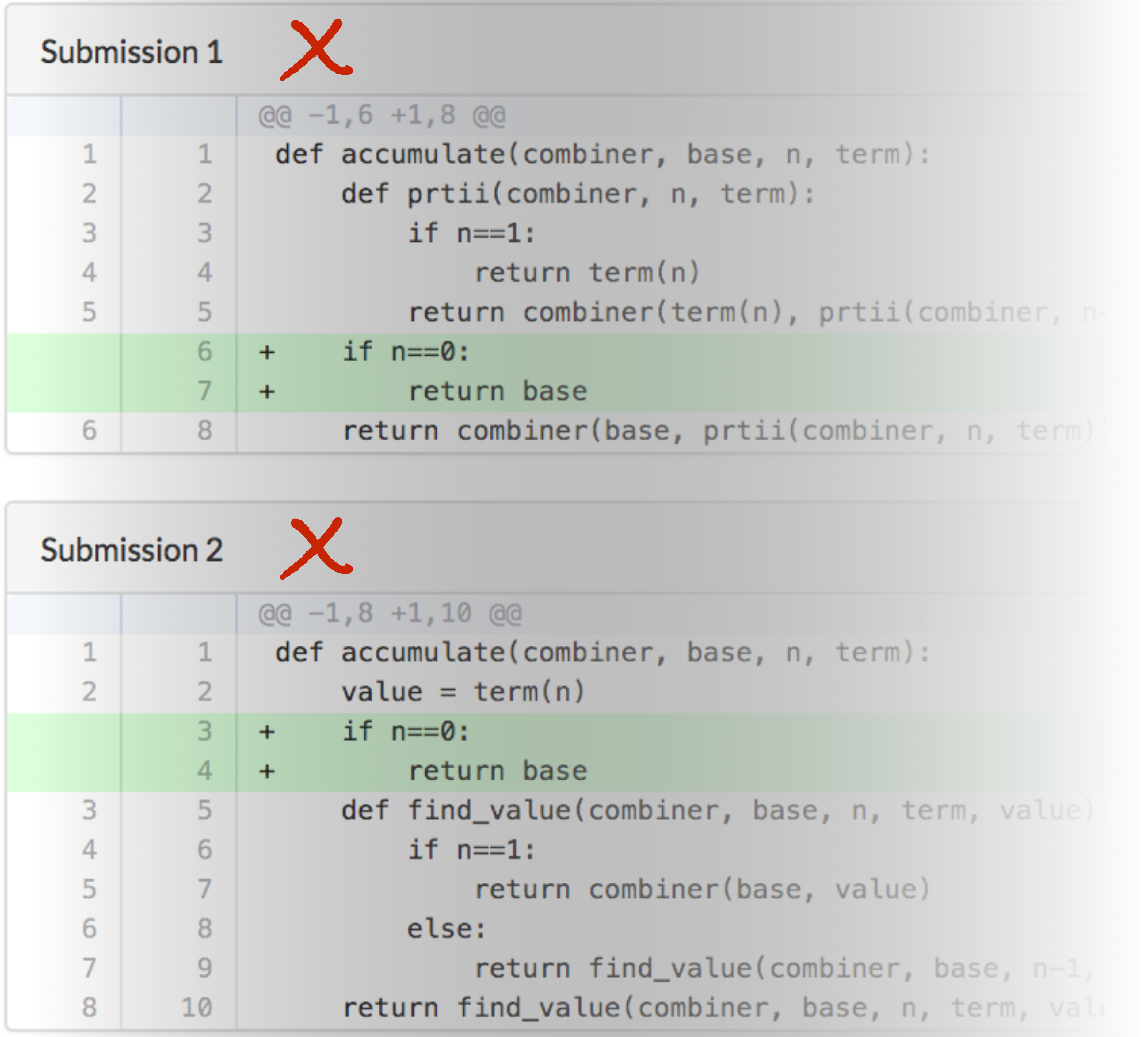
$$\textit{<exp>} * \textit{<name>} \xrightarrow{\text{Insert}} \textit{<exp>} \boxed{* \text{ term}(\textit{<name>})}$$

Learning Bug-Fixing Code Transformations



We Scale Up a Little Teacher-Written Feedback by Attaching It to Code Transformations

Incorrect Student Code Submissions



Submission 1 ✗

```
@@ -1,6 +1,8 @@
 def accumulate(combiner, base, n, term):
     def prtii(combiner, n, term):
         if n==1:
             return term(n)
         return combiner(term(n), prtii(combiner, n-
+    if n==0:
+        return base
     return combiner(base, prtii(combiner, n, term)
```

Submission 2 ✗

```
@@ -1,8 +1,10 @@
 def accumulate(combiner, base, n, term):
     value = term(n)
+    if n==0:
+        return base
     def find_value(combiner, base, n, term, value)
         if n==1:
             return combiner(base, value)
         else:
             return find_value(combiner, base, n-1,
     return find_value(combiner, base, n, term, val
```

Code Transformation (add base case)

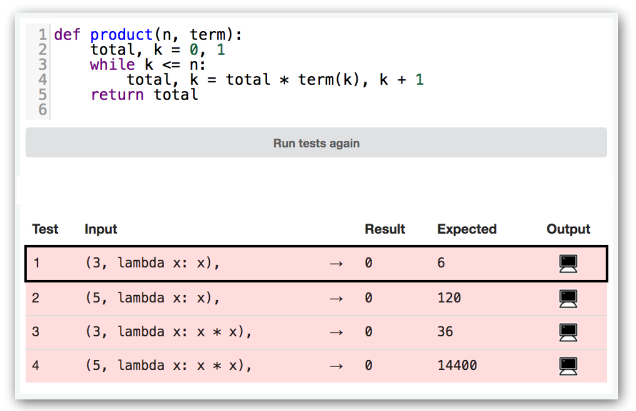
Teacher Comments

What happens when n is zero?
Hint: look at lecture 5's slides on base cases.

Two Interfaces for Attaching Feedback to Code Transformations

MistakeBrowser: giving feedback on clusters

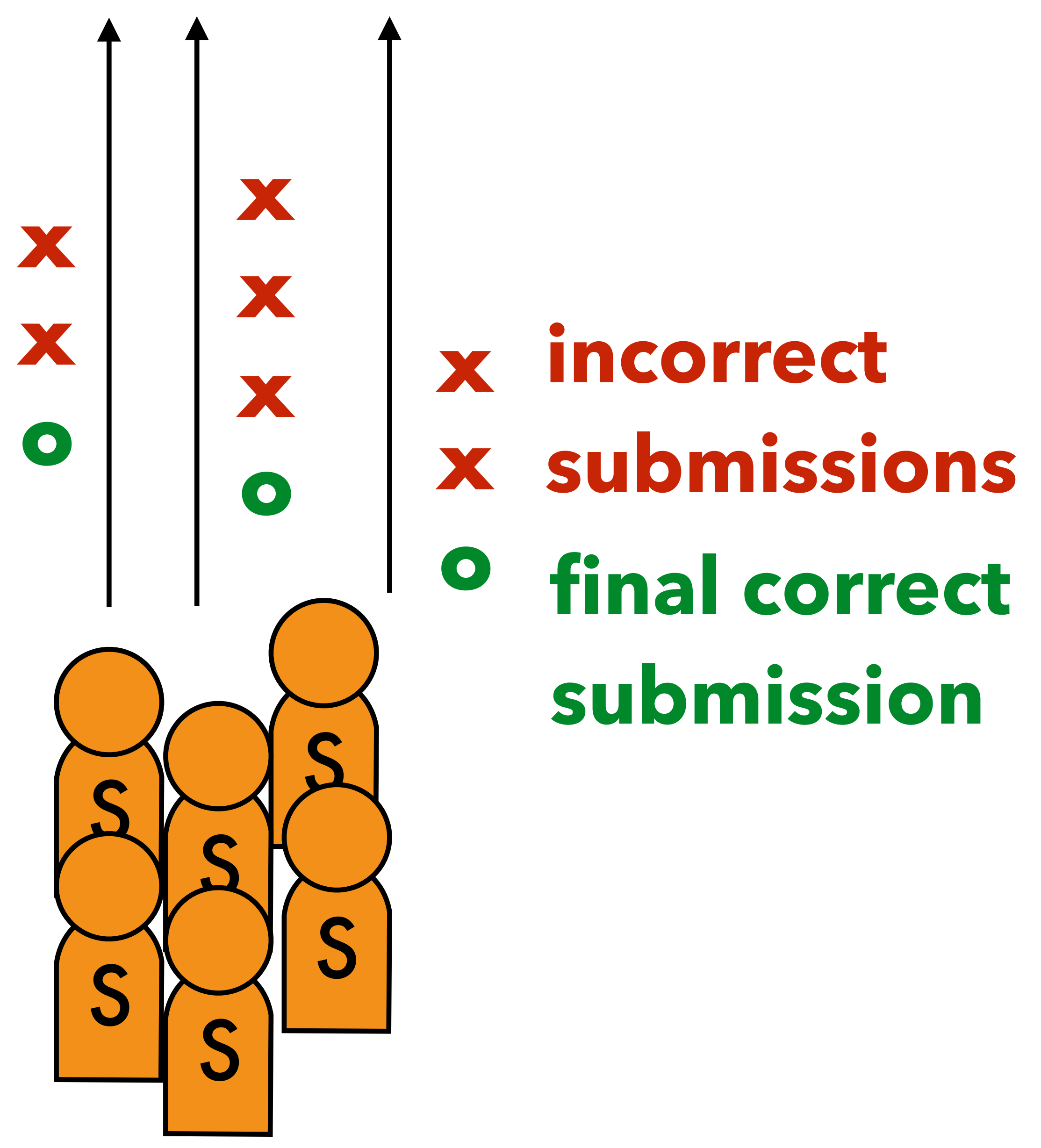
Learn transformations from Autograder



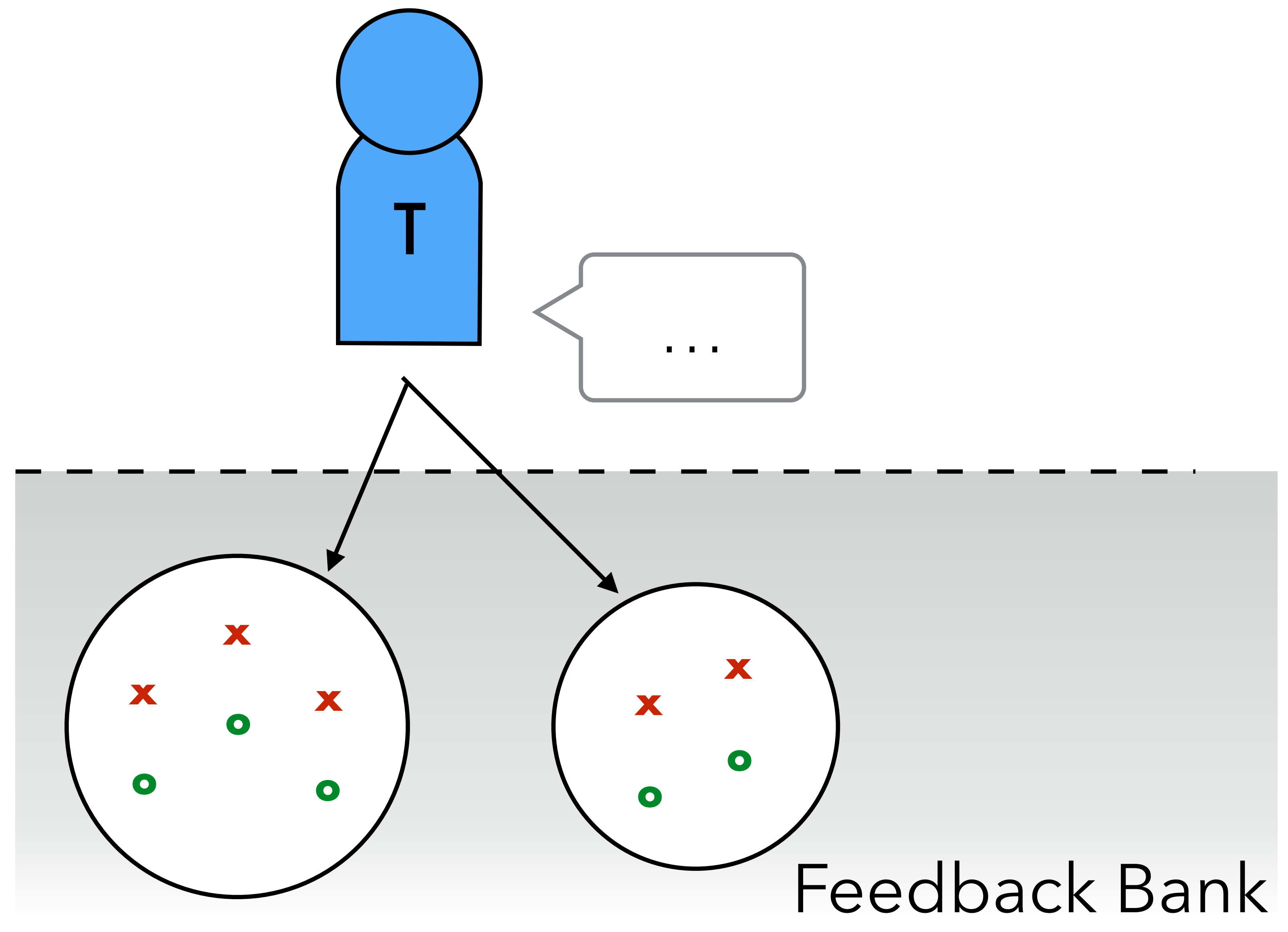
```
def product(n, term):
    total, k = 0, 1
    while k <= n:
        total, k = total * term(k), k + 1
    return total
```

Run tests again

Test	Input		Result	Expected	Output
1	(3, lambda x: x),	→	0	6	
2	(5, lambda x: x),	→	0	120	
3	(3, lambda x: x * x),	→	0	36	
4	(5, lambda x: x * x),	→	0	14400	



Collect feedback from teachers

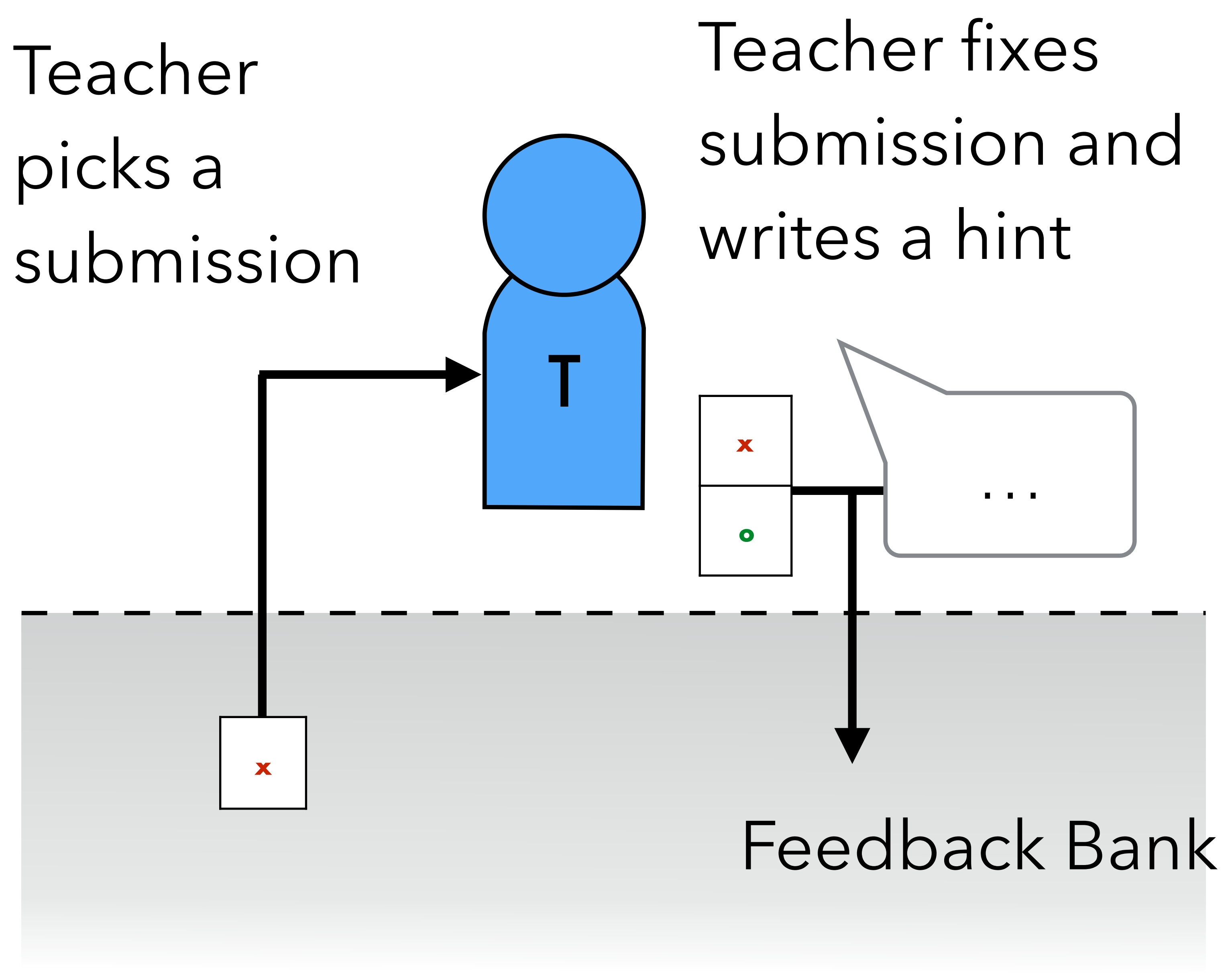


Related Systems: *Divide and Conquer* [ITS14], *AutoStyle* [ITS16]

Two Interfaces for Attaching Feedback to Code Transformations

FixPropagator: attaching feedback to individual fixes

Learns transformations from *and* collect feedback from…


Teacher picks a submission
T
Teacher fixes submission and writes a hint
…
Feedback Bank

Our Program Synthesis Backend

Refazer *(/hɛ.fa.ˈze(h)/)*

Means “To redo.”

Using *Refazer* [ICSE17] as a backend, our systems **learn bug-fixing code transformations.**

Contributions

- An approach for combining human expertise with program synthesis for delivering **reusable, scalable code feedback**

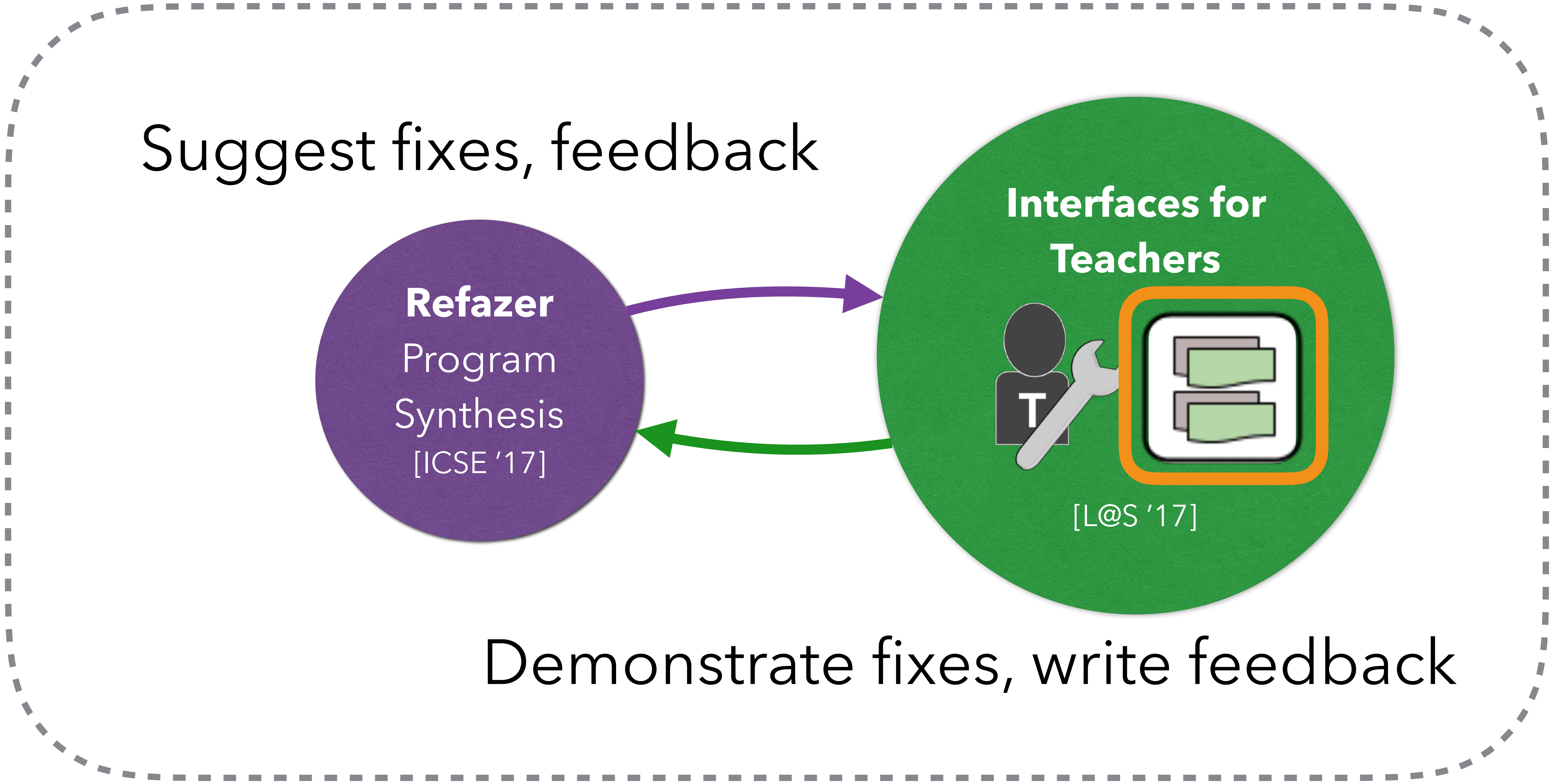
- Implementations of two different systems that use our approach: *FixPropagator*, *MistakeBrowser*

- In-lab studies that suggest that the systems fulfill our goals, also inform teachers about common student bugs

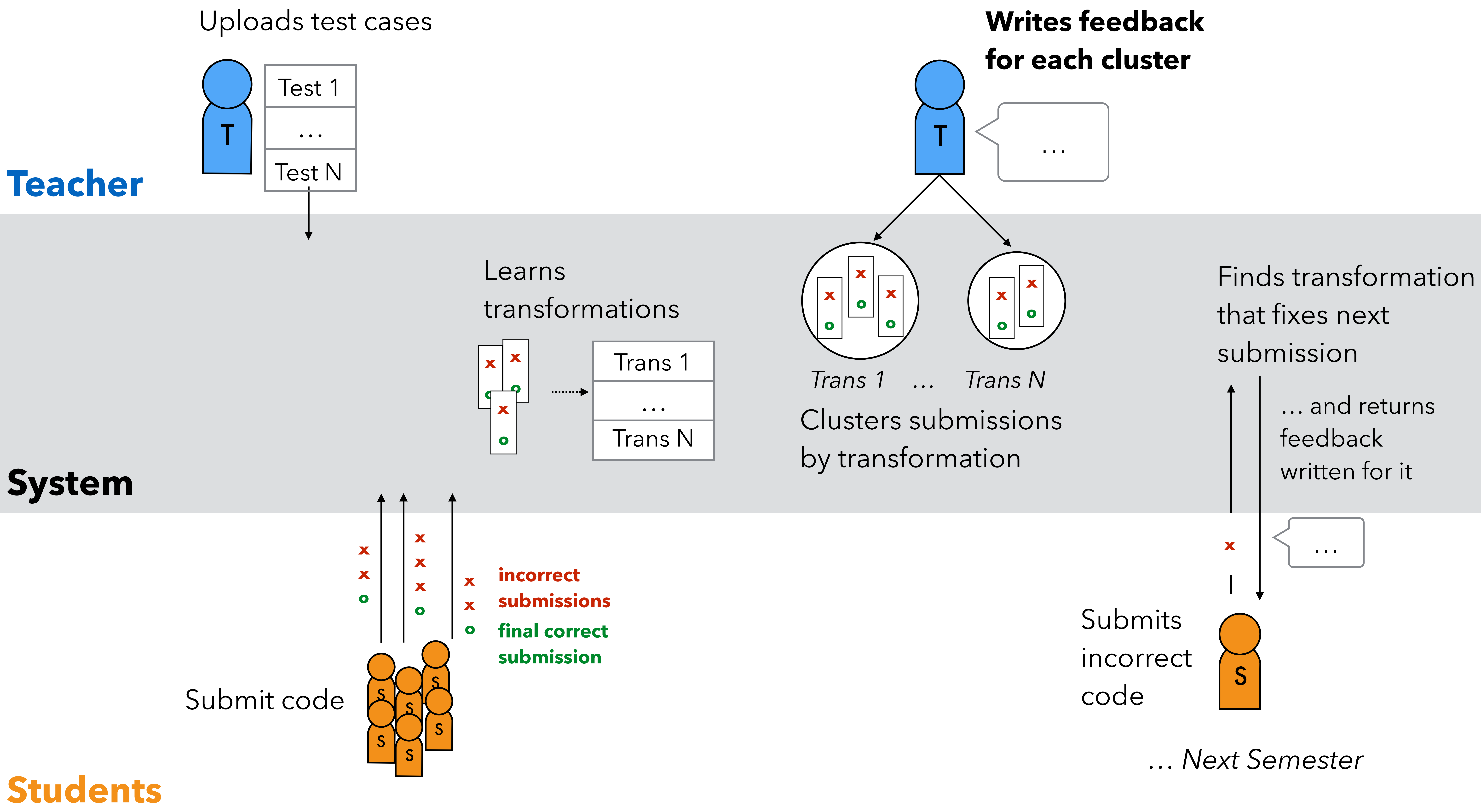
Outline

- ~~Related Work~~
- ~~Program Synthesis~~
- Systems
- Evaluation

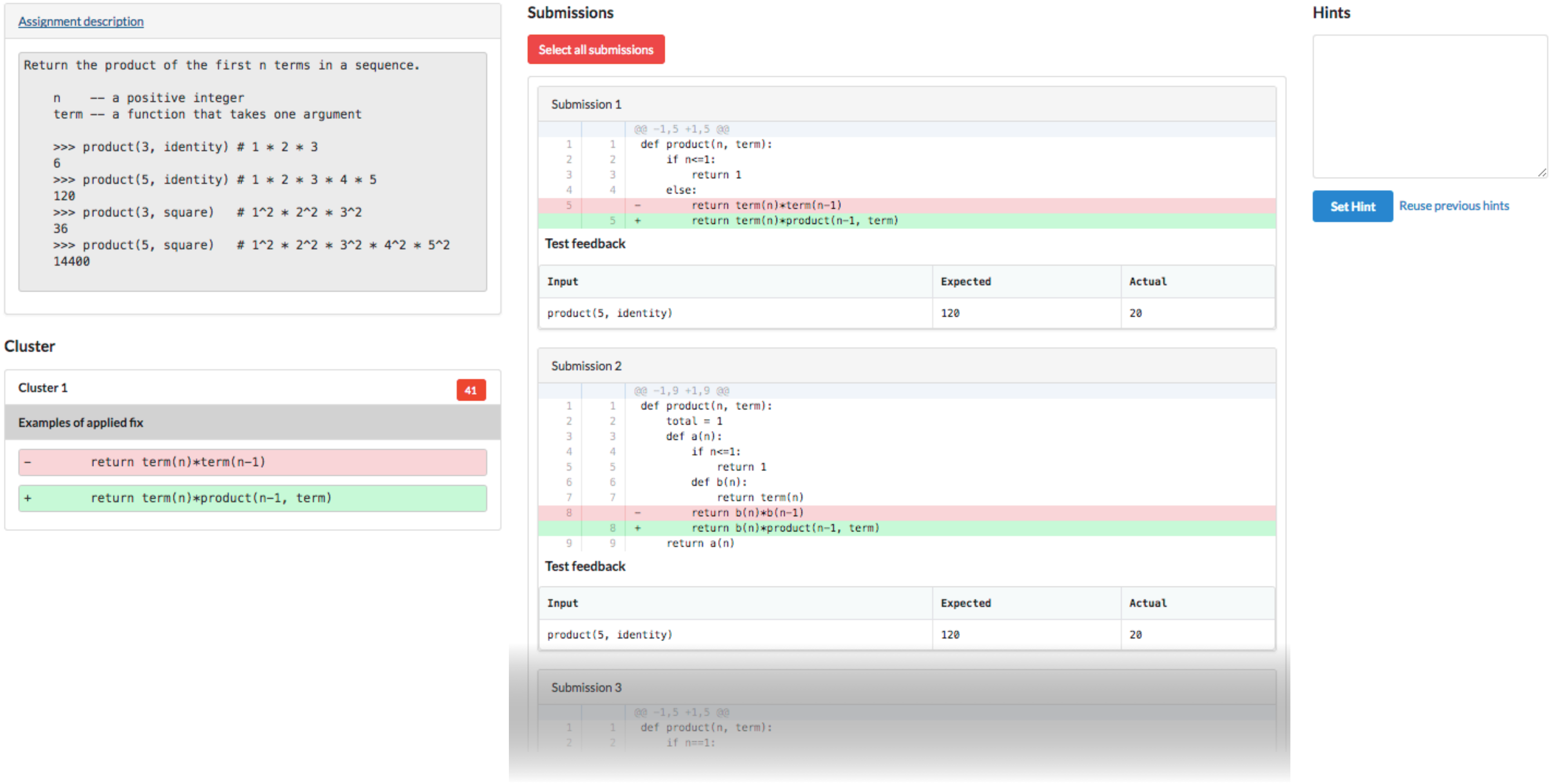
System Design



Mixed-initiative workflows



Systems: MistakeBrowser



Assignment description

```
Return the product of the first n terms in a sequence.

    n    -- a positive integer
    term -- a function that takes one argument

    >>> product(3, identity) # 1 * 2 * 3
    6
    >>> product(5, identity) # 1 * 2 * 3 * 4 * 5
    120
    >>> product(3, square)   # 1^2 * 2^2 * 3^2
    36
    >>> product(5, square)   # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
    14400
```

Cluster

Cluster 1 41

Examples of applied fix

```
-        return term(n)*term(n-1)
+        return term(n)*product(n-1, term)
```

Submissions

Select all submissions

Submission 1

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n<=1:
         return 1
     else:
-        return term(n)*term(n-1)
+        return term(n)*product(n-1, term)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

```
@@ -1,9 +1,9 @@
 def product(n, term):
     total = 1
     def a(n):
         if n<=1:
             return 1
         def b(n):
             return term(n)
-        return b(n)*b(n-1)
+        return b(n)*product(n-1, term)
     return a(n)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

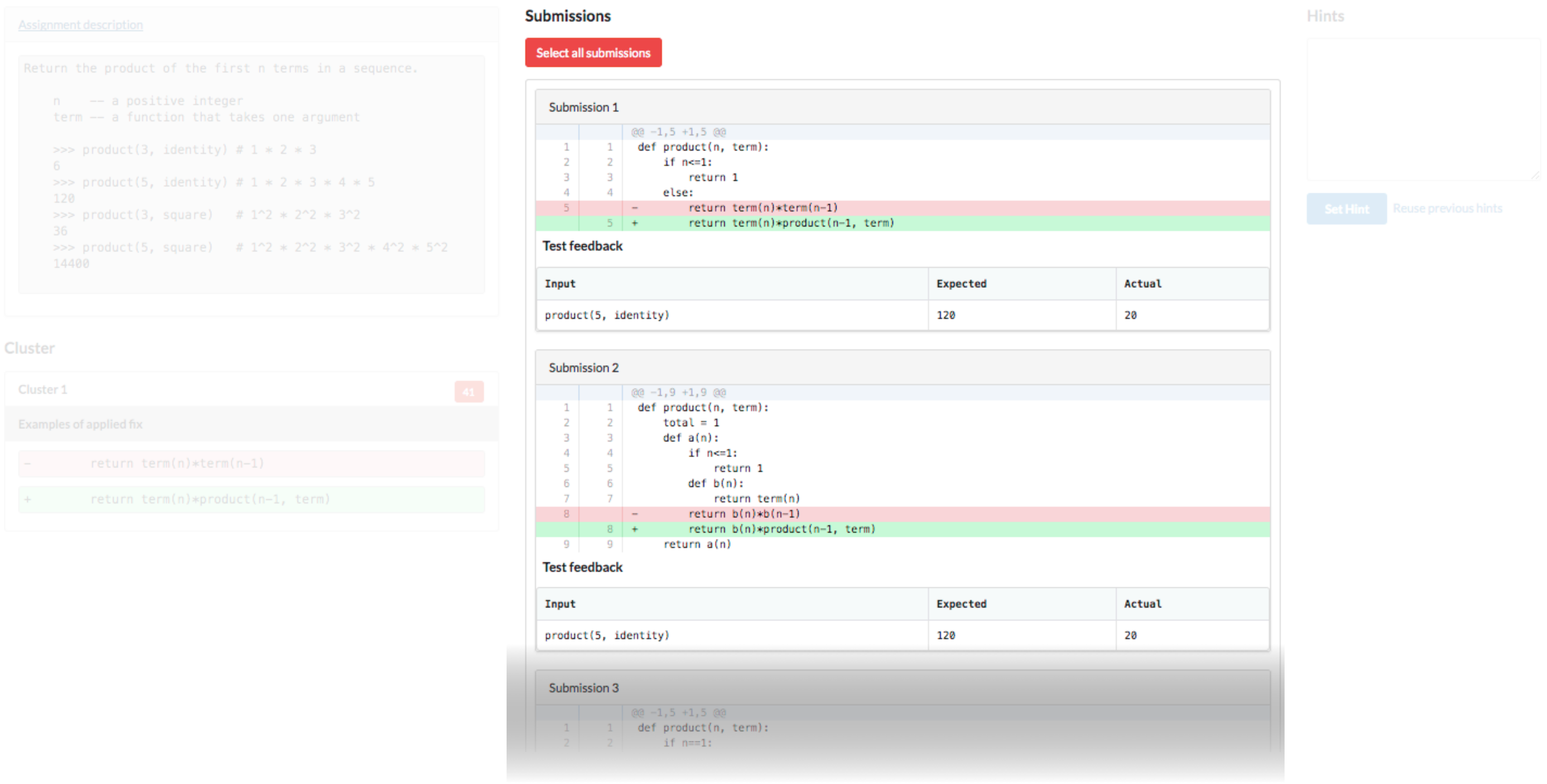
Submission 3

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n==1:
```

Hints

Set Hint   Reuse previous hints

Systems: MistakeBrowser



Assignment description

```
Return the product of the first n terms in a sequence.

    n    -- a positive integer
    term -- a function that takes one argument

    >>> product(3, identity) # 1 * 2 * 3
    6
    >>> product(5, identity) # 1 * 2 * 3 * 4 * 5
    120
    >>> product(3, square)   # 1^2 * 2^2 * 3^2
    36
    >>> product(5, square)   # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
    14400
```

Cluster

Cluster 1 41

Examples of applied fix

```
-        return term(n)*term(n-1)
+        return term(n)*product(n-1, term)
```

Submissions

Select all submissions

Submission 1

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n<=1:
         return 1
     else:
-        return term(n)*term(n-1)
+        return term(n)*product(n-1, term)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

```
@@ -1,9 +1,9 @@
 def product(n, term):
     total = 1
     def a(n):
         if n<=1:
             return 1
         def b(n):
             return term(n)
-        return b(n)*b(n-1)
+        return b(n)*product(n-1, term)
     return a(n)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

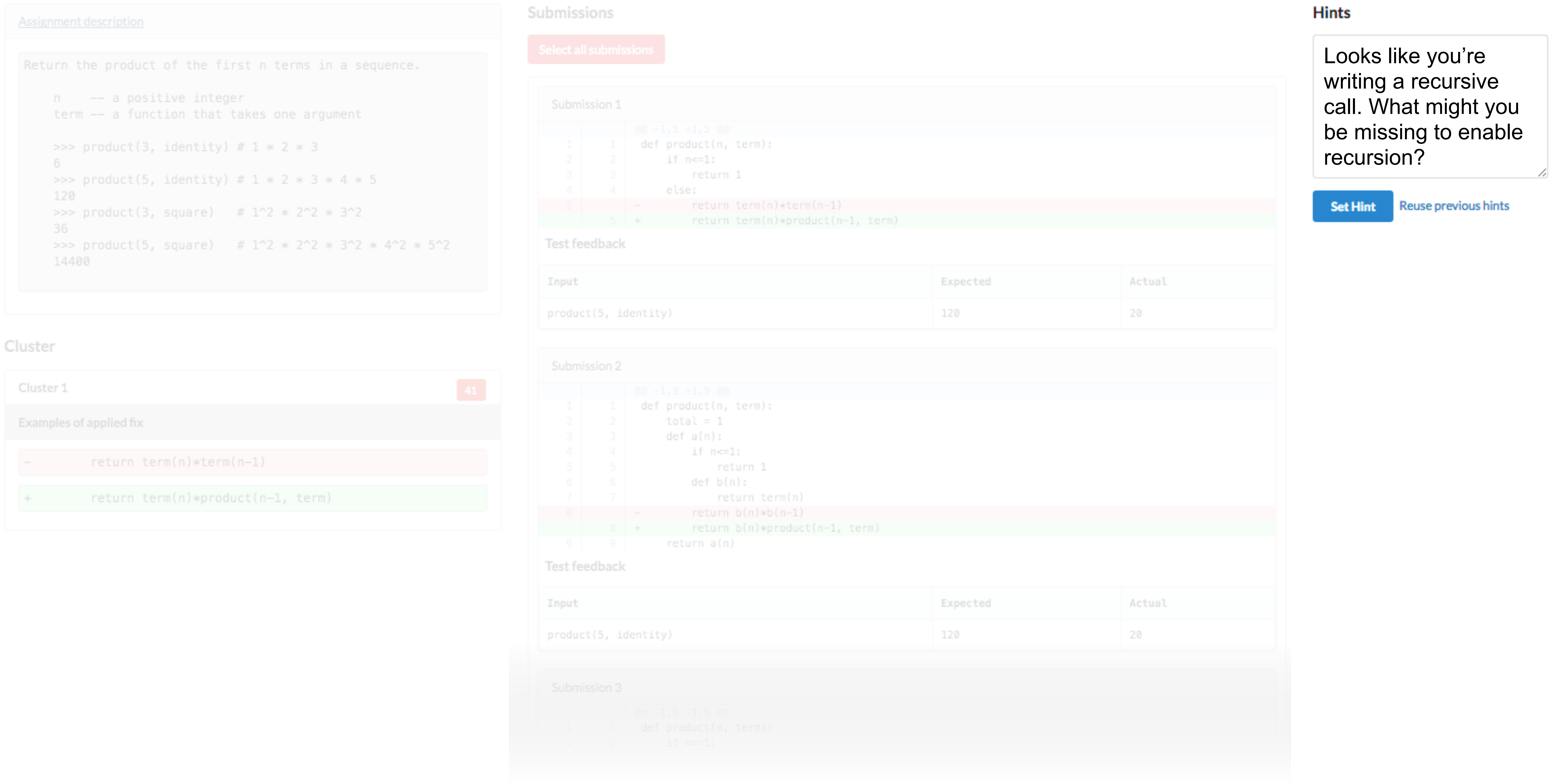
Submission 3

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n==1:
```

Hints

Set Hint Reuse previous hints

Systems: MistakeBrowser



Assignment description

```
Return the product of the first n terms in a sequence.

    n    -- a positive integer
    term -- a function that takes one argument

    >>> product(3, identity) # 1 * 2 * 3
    6
    >>> product(5, identity) # 1 * 2 * 3 * 4 * 5
    120
    >>> product(3, square)   # 1^2 * 2^2 * 3^2
    36
    >>> product(5, square)   # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
    14400
```

Cluster

Cluster 1 41

Examples of applied fix

```
-       return term(n)*term(n-1)
+       return term(n)*product(n-1, term)
```

Submissions

Select all submissions

Submission 1

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n<=1:
         return 1
     else:
-        return term(n)*term(n-1)
+        return term(n)*product(n-1, term)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 2

```
@@ -1,9 +1,9 @@
 def product(n, term):
     total = 1
     def a(n):
         if n<=1:
             return 1
         def b(n):
             return term(n)
-        return b(n)*b(n-1)
+        return b(n)*product(n-1, term)
     return a(n)
```

Test feedback

Input	Expected	Actual
product(5, identity)	120	20

Submission 3

```
@@ -1,5 +1,5 @@
 def product(n, term):
     if n==1:
```

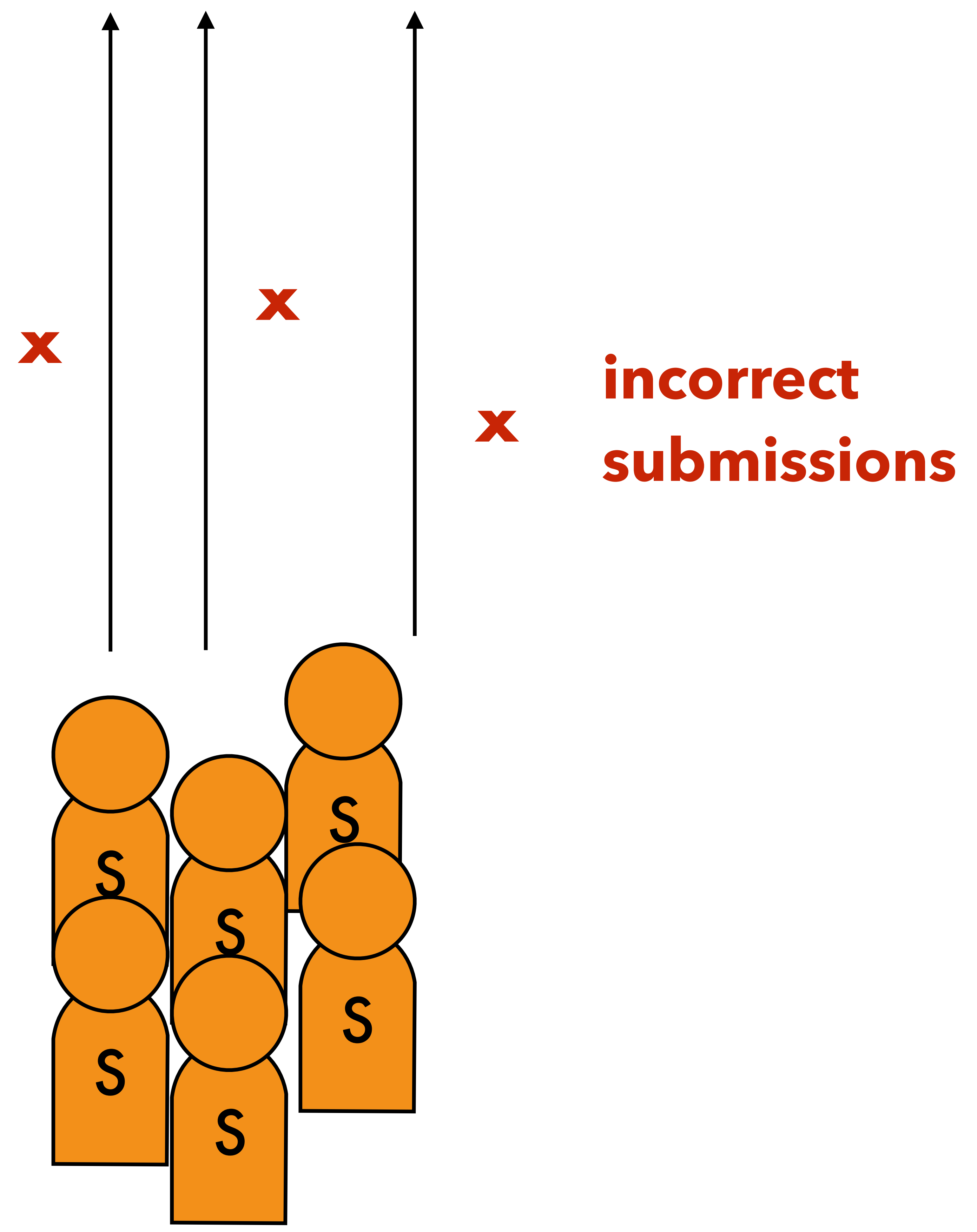
Hints

Looks like you're writing a recursive call. What might you be missing to enable recursion?

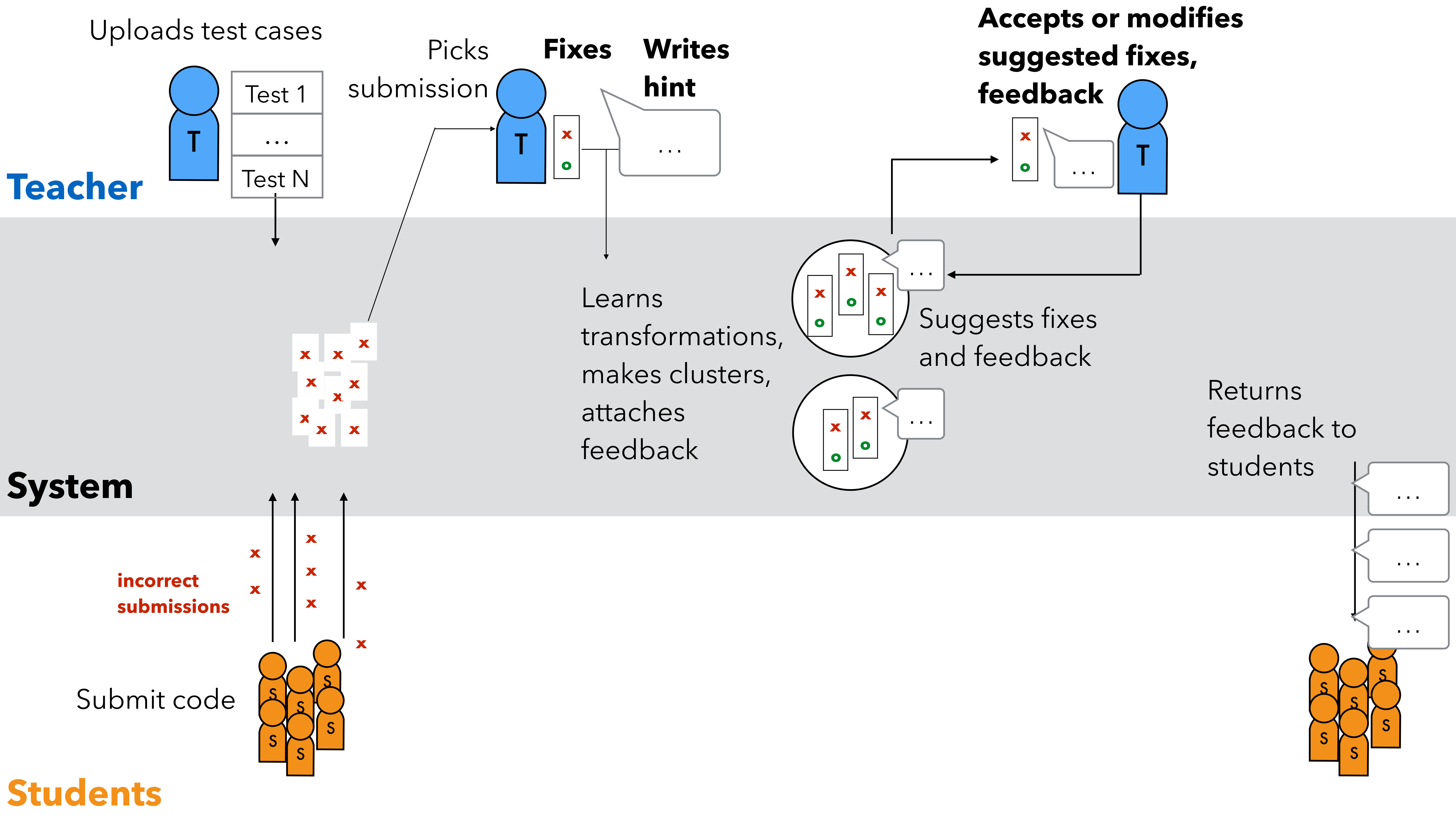
Set Hint   Reuse previous hints

Systems: MistakeBrowser

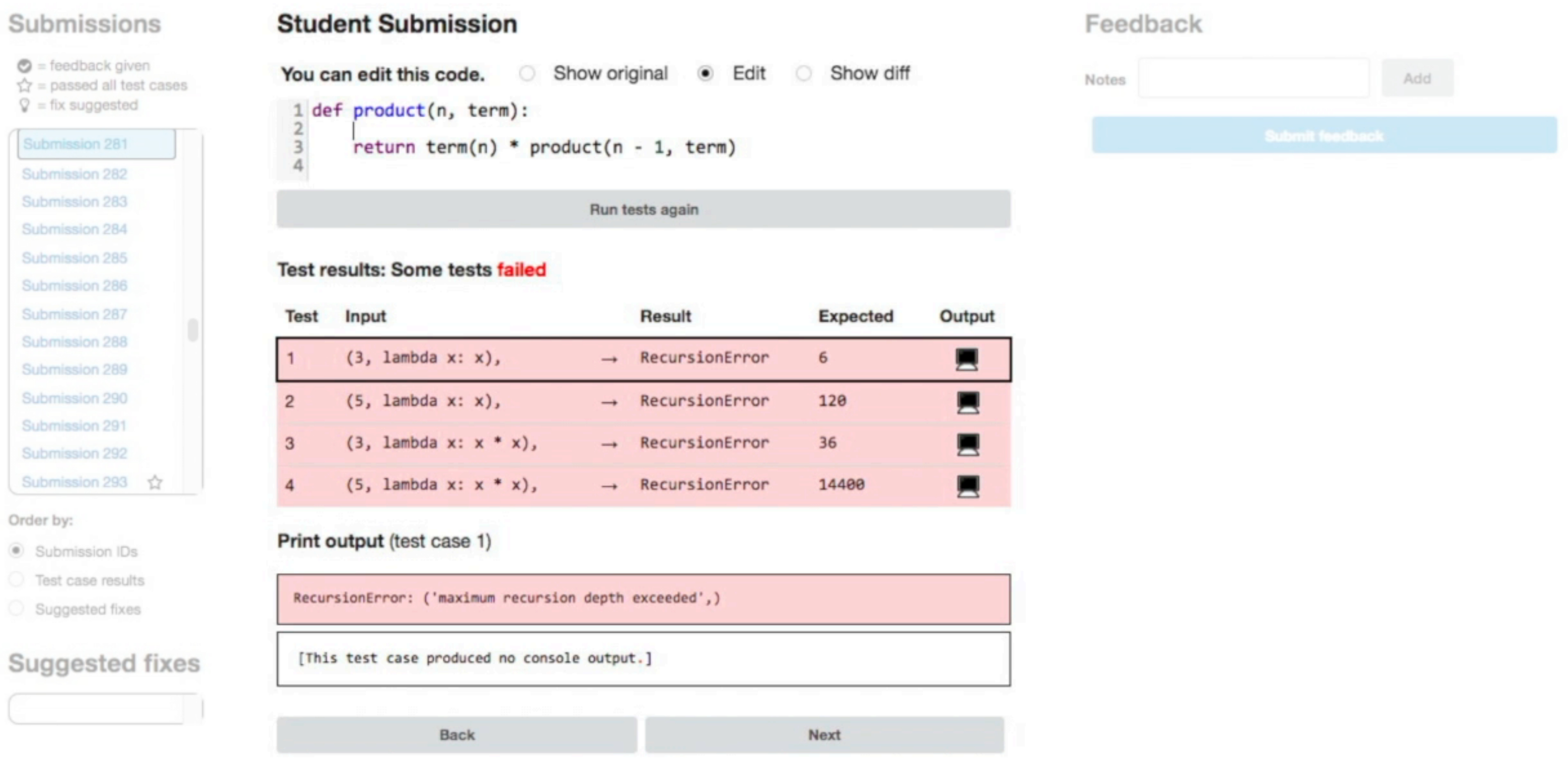
But Not All Classes Have Submission Histories for Hundreds of Students



Submit code


Uploads test cases
Test 1
...
Test N
Teacher
Picks submission
Fixes
Writes hint
...
Accepts or modifies suggested fixes, feedback
...
Learns transformations, makes clusters, attaches feedback
...
Suggests fixes and feedback
...
Returns feedback to students
...
...
...
System
incorrect submissions
Submit code
Students

Systems: FixPropagator



Submissions

✓ = feedback given
☆ = passed all test cases
💡 = fix suggested

Submission 281
Submission 282
Submission 283
Submission 284
Submission 285
Submission 286
Submission 287
Submission 288
Submission 289
Submission 290
Submission 291
Submission 292
Submission 293 ☆

Order by:
[✓ Submission IDs]
Test case results
Suggested fixes

Suggested fixes

Student Submission

You can edit this code. Show original [✓ Edit] Show diff

```
def product(n, term):

    return term(n) * product(n - 1, term)

```

Run tests again

Test results: Some tests failed

Test	Input		Result	Expected	Output
1	(3, lambda x: x),	→	RecursionError	6	■
2	(5, lambda x: x),	→	RecursionError	120	■
3	(3, lambda x: x * x),	→	RecursionError	36	■
4	(5, lambda x: x * x),	→	RecursionError	14400	■

Print output (test case 1)

```
RecursionError: ('maximum recursion depth exceeded',)
```

```
[This test case produced no console output.]
```

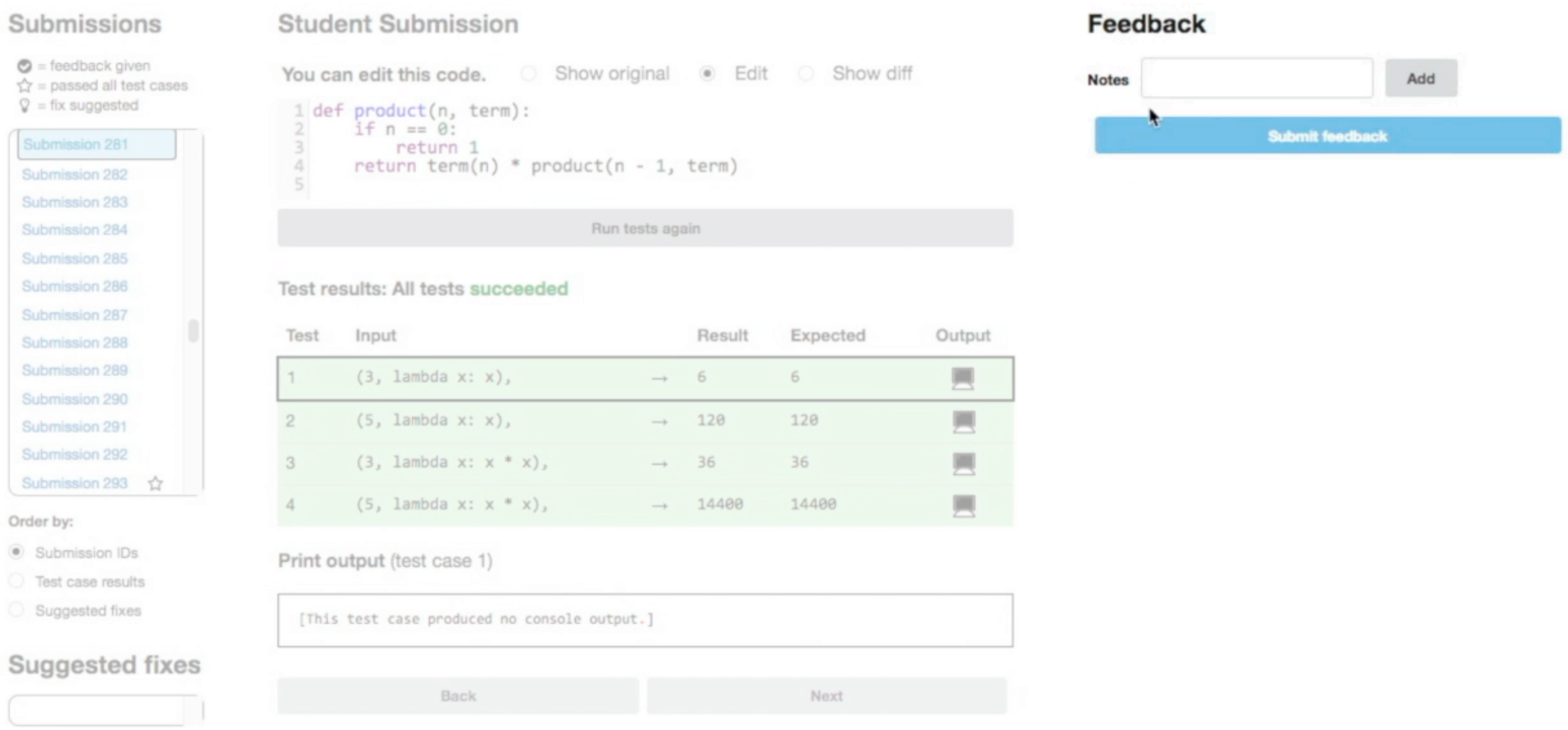
Back Next

Feedback

Notes Add

Submit feedback

Systems: FixPropagator



Submissions

● = feedback given
☆ = passed all test cases
💡 = fix suggested

Submission 281
Submission 282
Submission 283
Submission 284
Submission 285
Submission 286
Submission 287
Submission 288
Submission 289
Submission 290
Submission 291
Submission 292
Submission 293 ☆

Order by:
[✓ Submission IDs]
Test case results
Suggested fixes

Suggested fixes

Student Submission

You can edit this code. Show original [✓ Edit] Show diff

```
def product(n, term):
    if n == 0:
        return 1
    return term(n) * product(n - 1, term)

```

Run tests again

Test results: All tests succeeded

Test	Input		Result	Expected	Output
1	`(3, lambda x: x),`	→	6	6	
2	`(5, lambda x: x),`	→	120	120	
3	`(3, lambda x: x * x),`	→	36	36	
4	`(5, lambda x: x * x),`	→	14400	14400	

Print output (test case 1)

```
[This test case produced no console output.]
```

Back Next

Feedback

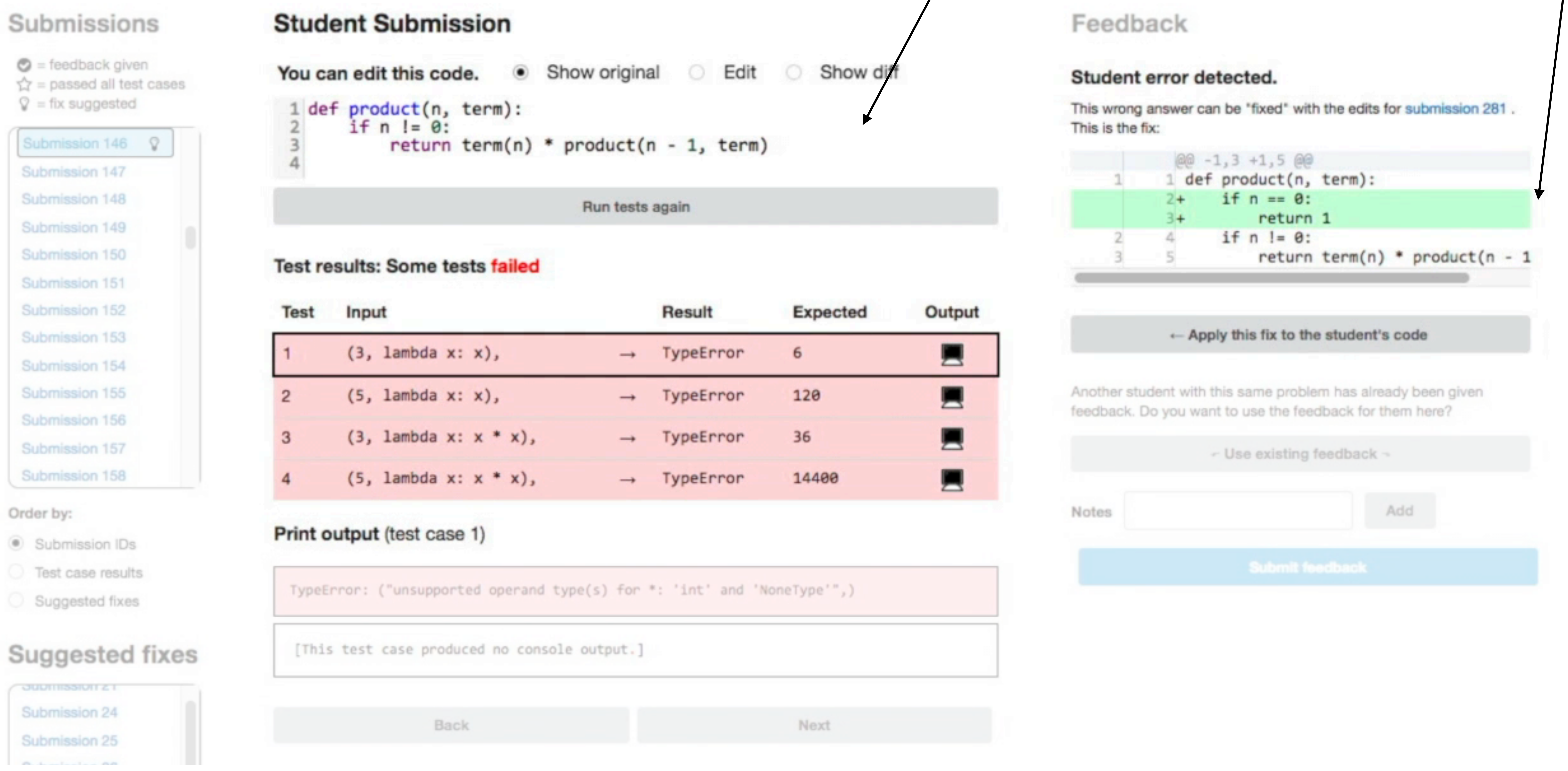
Notes Add

Submit feedback

Systems: FixPropagator

New Student Submission with Same Bug

Suggested Fix



Submissions

✓ = feedback given
☆ = passed all test cases
💡 = fix suggested

- Submission 146 💡
- Submission 147
- Submission 148
- Submission 149
- Submission 150
- Submission 151
- Submission 152
- Submission 153
- Submission 154
- Submission 155
- Submission 156
- Submission 157
- Submission 158

Order by:

[✓ Submission IDs] Test case results Suggested fixes

Suggested fixes

- Submission 21
- Submission 24
- Submission 25

Student Submission

You can edit this code. [✓ Show original] Edit Show diff

```
def product(n, term):
    if n != 0:
        return term(n) * product(n - 1, term)
```

Run tests again

Test results: Some tests failed

Test	Input		Result	Expected	Output
1	`(3, lambda x: x),`	→	`TypeError`	`6`	■
2	`(5, lambda x: x),`	→	`TypeError`	`120`	■
3	`(3, lambda x: x * x),`	→	`TypeError`	`36`	■
4	`(5, lambda x: x * x),`	→	`TypeError`	`14400`	■

Print output (test case 1)

```
TypeError: ("unsupported operand type(s) for *: 'int' and 'NoneType'",)
```

```
[This test case produced no console output.]
```

Back Next

Feedback

Student error detected.

This wrong answer can be "fixed" with the edits for submission 281 .
This is the fix:

```
@@ -1,3 +1,5 @@
 def product(n, term):
+    if n == 0:
+        return 1
     if n != 0:
         return term(n) * product(n - 1
```

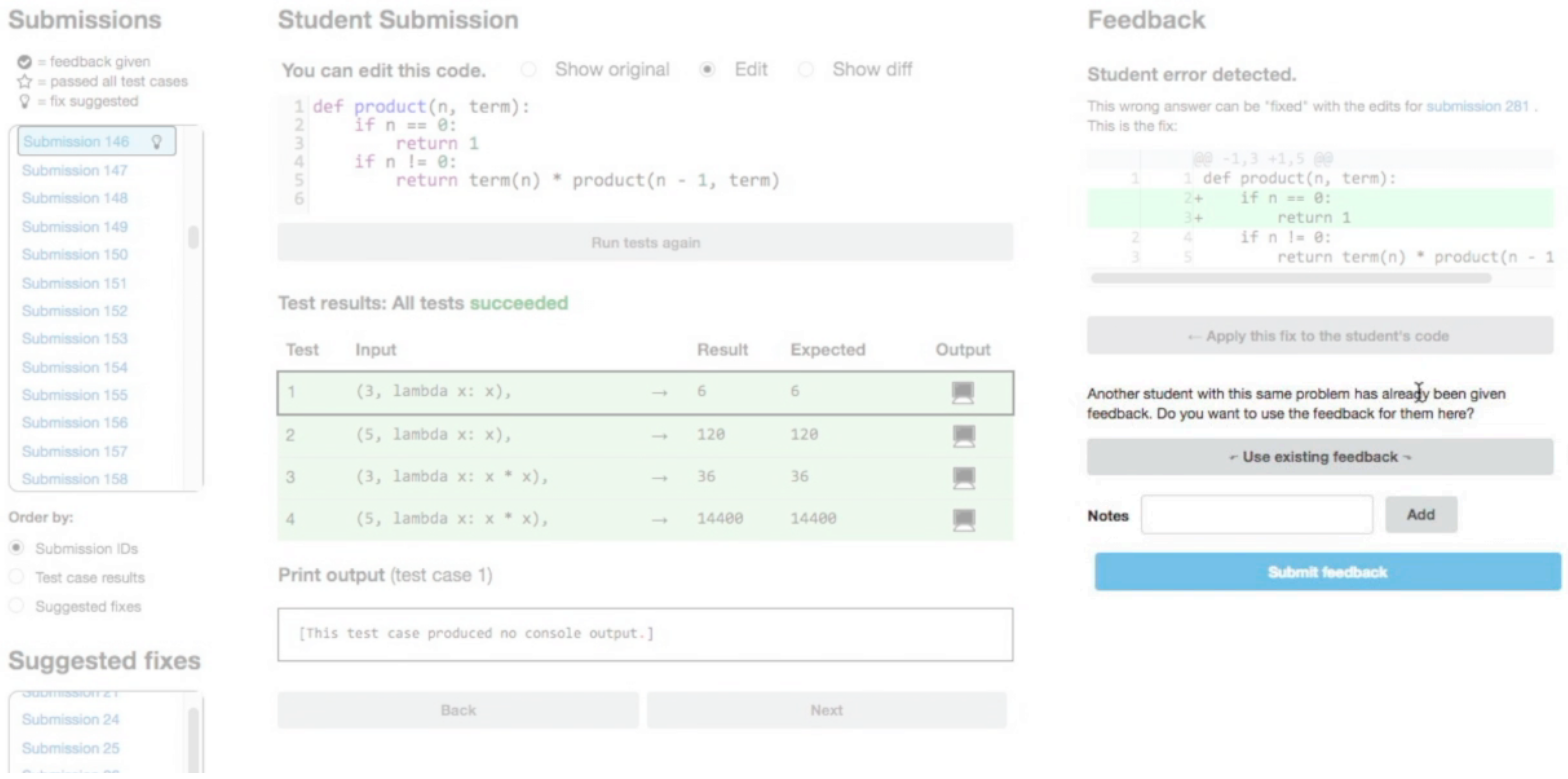
← Apply this fix to the student's code

Another student with this same problem has already been given feedback. Do you want to use the feedback for them here?

← Use existing feedback →

Notes Add

Submit feedback



Order by: [✓ Submission IDs] Test case results Suggested fixes

Student Submission

You can edit this code. Show original [✓ Edit] Show diff

```
def product(n, term):
    if n == 0:
        return 1
    if n != 0:
        return term(n) * product(n - 1, term)
```

Test results: All tests succeeded

Test	Input		Result	Expected	Output
1	(3, lambda x: x),	→	6	6	
2	(5, lambda x: x),	→	120	120	
3	(3, lambda x: x * x),	→	36	36	
4	(5, lambda x: x * x),	→	14400	14400	

Print output (test case 1)

```
[This test case produced no console output.]
```

Feedback

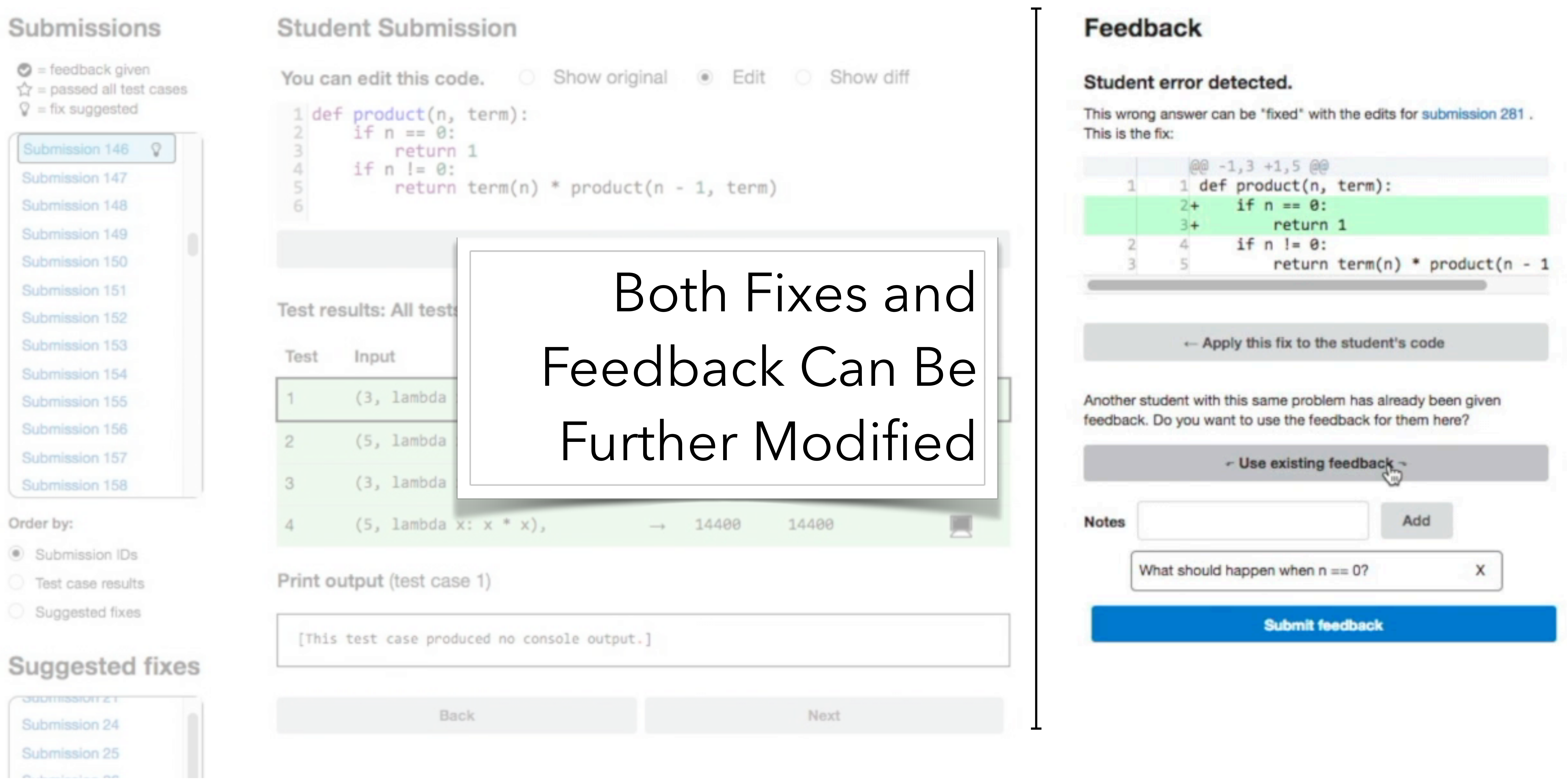
Student error detected.

This wrong answer can be "fixed" with the edits for submission 281 .
This is the fix:

```
@@ -1,3 +1,5 @@
 def product(n, term):
+    if n == 0:
+        return 1
     if n != 0:
         return term(n) * product(n - 1
```

Another student with this same problem has already been given feedback. Do you want to use the feedback for them here?

Systems: FixPropagator


Submissions

✓ = feedback given
☆ = passed all test cases
💡 = fix suggested

- Submission 146 💡
- Submission 147
- Submission 148
- Submission 149
- Submission 150
- Submission 151
- Submission 152
- Submission 153
- Submission 154
- Submission 155
- Submission 156
- Submission 157
- Submission 158

Order by:
- [✓ Submission IDs]
- Test case results
- Suggested fixes

Suggested fixes

- Submission 21
- Submission 24
- Submission 25

Student Submission

You can edit this code. ○ Show original [✓ Edit] ○ Show diff

```
def product(n, term):
    if n == 0:
        return 1
    if n != 0:
        return term(n) * product(n - 1, term)
```

Test results: All tests

Test	Input			
1	(3, lambda			
2	(5, lambda			
3	(3, lambda			
4	(5, lambda x: x * x),	→	14400	14400

Print output (test case 1)

```
[This test case produced no console output.]
```

Back Next

Both Fixes and Feedback Can Be Further Modified

Feedback

Student error detected.

This wrong answer can be "fixed" with the edits for submission 281 . This is the fix:

```
@@ -1,3 +1,5 @@
 def product(n, term):
+    if n == 0:
+        return 1
     if n != 0:
         return term(n) * product(n - 1
```

← Apply this fix to the student's code

Another student with this same problem has already been given feedback. Do you want to use the feedback for them here?

← Use existing feedback →

Notes [ ] Add

What should happen when n == 0? X

Submit feedback

A Study of the Systems

Participants: Current and former teaching staff from CS1

MistakeBrowser (*N* = 9) FixPropagator (*N* = 8)

Interface Walkthrough (5 mins.)

Main Task (30 mins.): Giving feedback on student submissions
Measurements: Feedback, Manual corrections, Response to feedback recommendations (accepted, changed, rejected), Between-task surveys…

Qualitative Feedback: Survey and Post-interview

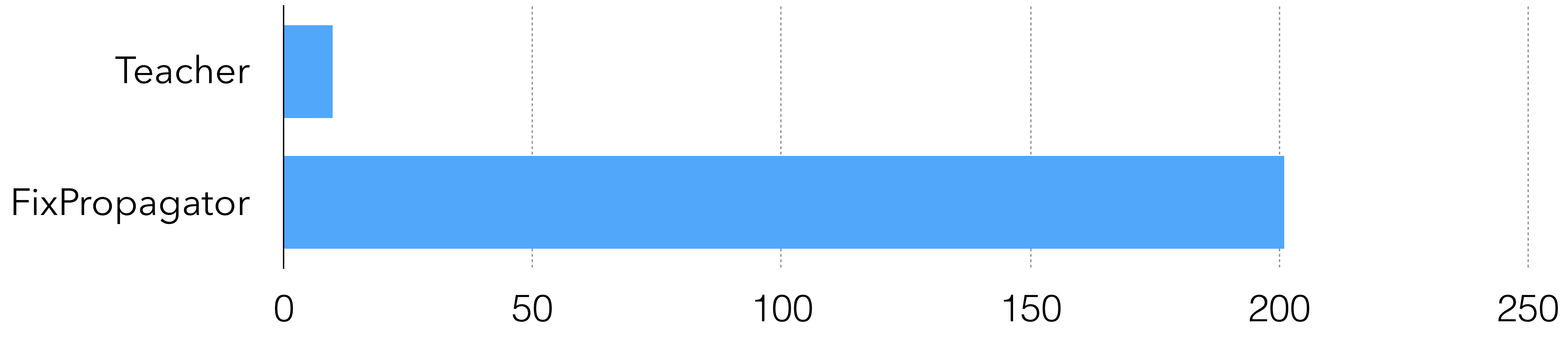
1. Can a **few manual corrections fix many** submissions?

1. Can a few manual corrections fix many submissions?

FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.

FixPropagator propagates fixes from dozens of corrections to hundreds of submissions.

Median # submissions given feedback by...



- Fixes were propagated within minutes (*median* = 2m20s, *σ* = 7m34s for each correction).

2. How often is a teacher’s **feedback relevant when it is matched** to other students’ submission?

2. How often is a teacher's feedback relevant when it is matched to other students' submission?

Feedback propagated with FixPropagator was correct a majority of the time, but not always.

Teachers reused feedback a median of 20 times, modifying it a median of 6 times (30%).

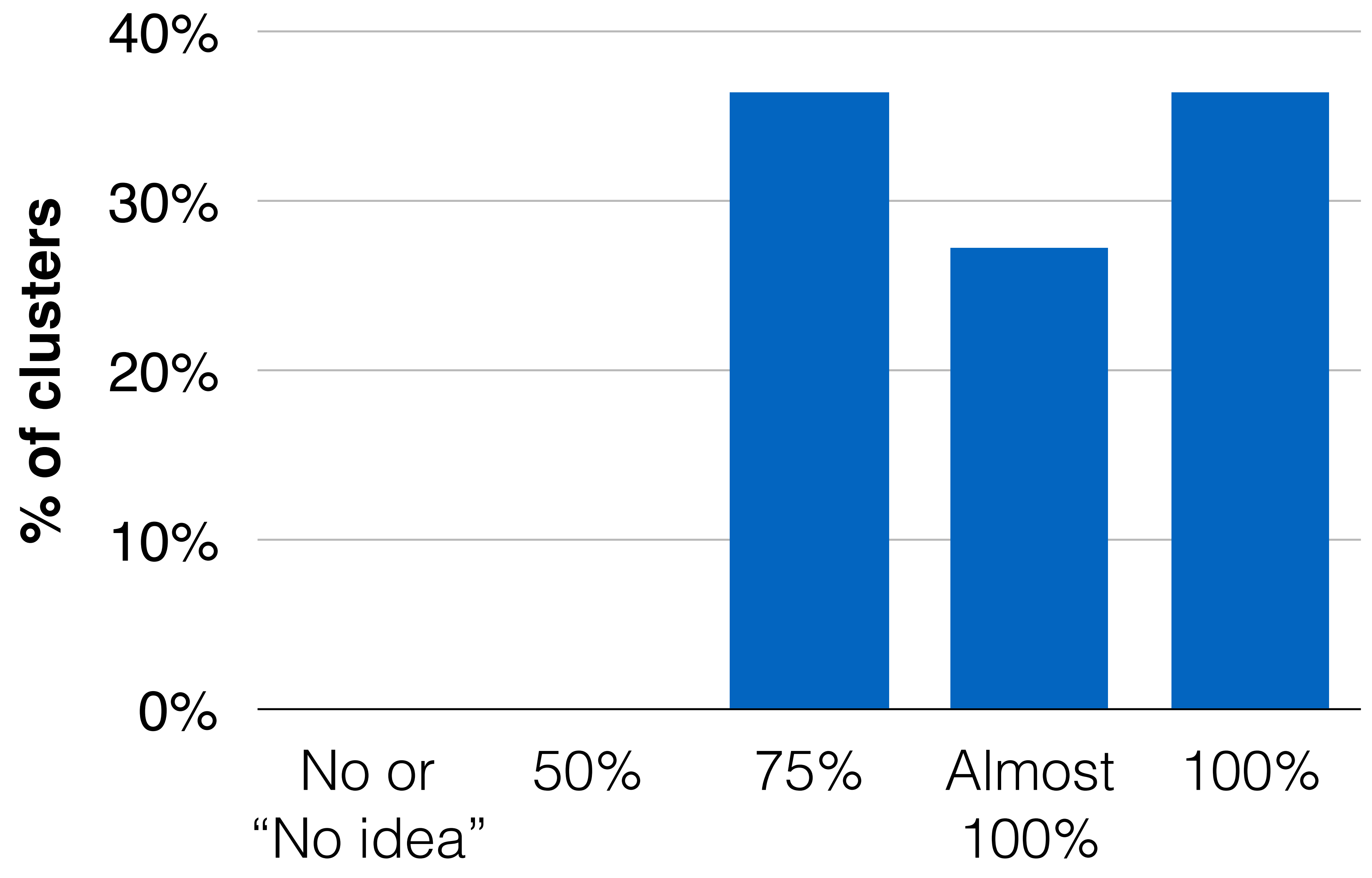
Generalizable Comment	Non-Generalizable Comment
"Check if you have the product of the correct number of terms."	"Your starting value of *z* should be a function, not an int."

2. How often is a teacher's feedback relevant when it is matched to other students' submission?

MistakeBrowser created conceptually consistent clusters of student bugs.

2. How often is a teacher's feedback relevant when it is matched to other students' submission?

MistakeBrowser created conceptually consistent clusters of student bugs.



Do these submissions share the same misconception?
Responses for *N* = 11 clusters

Evaluation Questions

1. Can a **few manual corrections fix many** submissions?

   With a median of 10 corrections, FixPropagator suggested fixes for a median of 201 submissions.

2. How often is a teacher's **feedback relevant** when it is matched to another student submission?

   Matched feedback was relevant ~75% of the time.

Limitations

- The impact of teacher feedback on student learning outcomes has not been evaluated
- Code transformations were created that fix submissions one or two bugs away from correct

Conclusion

We present an approach for combining human expertise with program synthesis for delivering reusable, scalable code feedback.

And two systems implementing this approach:

MistakeBrowser FixPropagator

Conclusion

We present an approach for combining human expertise with program synthesis for delivering reusable, scalable code feedback.

And two systems implementing this approach:

MistakeBrowser FixPropagator

Questions?