https://vintagecomputer.ca/lear-siegler-adm-3a-terminal/

CSE 13S Spring 2023

Computer Systems and C Programming

Lear Siegler ADM-3A Computer Terminal: Possibly the origin of vi's use of HJKL for cursor movement.


Classroom information
Class time and location
M/W/F from 9:20 am – 10:25 am
Performing Arts M110 (Media Theater)
Final-exam day/time
Monday, June 12, 8:00 am – 11:00 am

Instructor

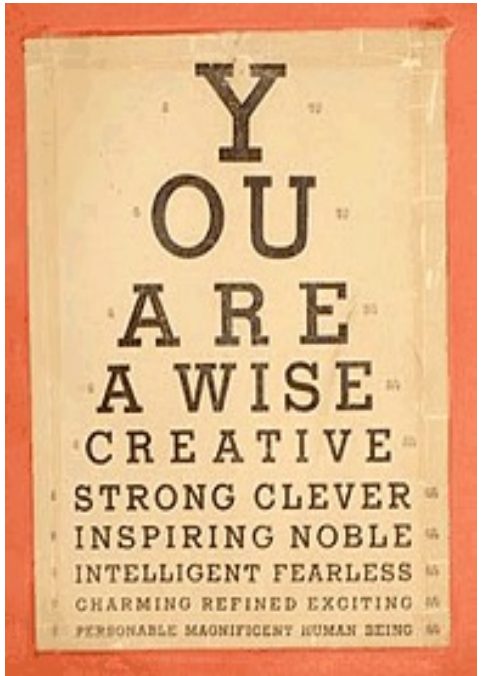
Dr. Kerry Veenstra
veenstra@ucsc.edu

Engineering 2 Building, Room 247A
(this is a shared office)

Office hours:
Tuesday 10:30 am – 12:30 pm
Thursday 2:00 pm – 4:00 pm

I'm totally supportive of DRC accommodations

- Bring me or email me your form ASAP
- Some folks need accommodations for the final only, some may need something for the quizzes: if so, we need to talk SOON!



So where does your grade come from?

- 20% Quizzes (top $n-1$ scores)
  - In class every Friday
  - I drop your lowest quiz score
- 50% Programming Assignments
- 30% Final Exam

I record the classes and post slides. **You** choose if you come to lecture—except for the quizzes.

NOTE: Assigned seats for the final exam

Canvas Web Site

- https://canvas.ucsc.edu/courses/62884

- Staff & Schedules (*still* under construction)
  - Office Hours
  - Discussion Section Times
  - Tutors & Times

Painless Way to Learn a Programming Language

Write a series of tiny programs to verify your understanding of what you read.

git: Don't Commit "Derived" Files

Name	Last commit	Last update
..		
CHEATING.pdf	Replace CHEATING.pdf	2 days ago
README.md	modified readme	11 hours ago
essay.pdf	added essay.pdf	10 hours ago
hello	added hello.c	2 days ago
hello.c	clang formatted hello.c	11 hours ago

Don't commit an executable file.
Commit the .c and .h files from which it is generated.

Useful git Commands

git add *file*	**("track" *file*)**
git status	**(what is tracked?)**
git commit -m "comment"	**("tracked" files into repo)**
git ls-files	**(list files in local repo)**
git rm --cached *file*	**(remove file from local repo)**

Comic by xkcd.com

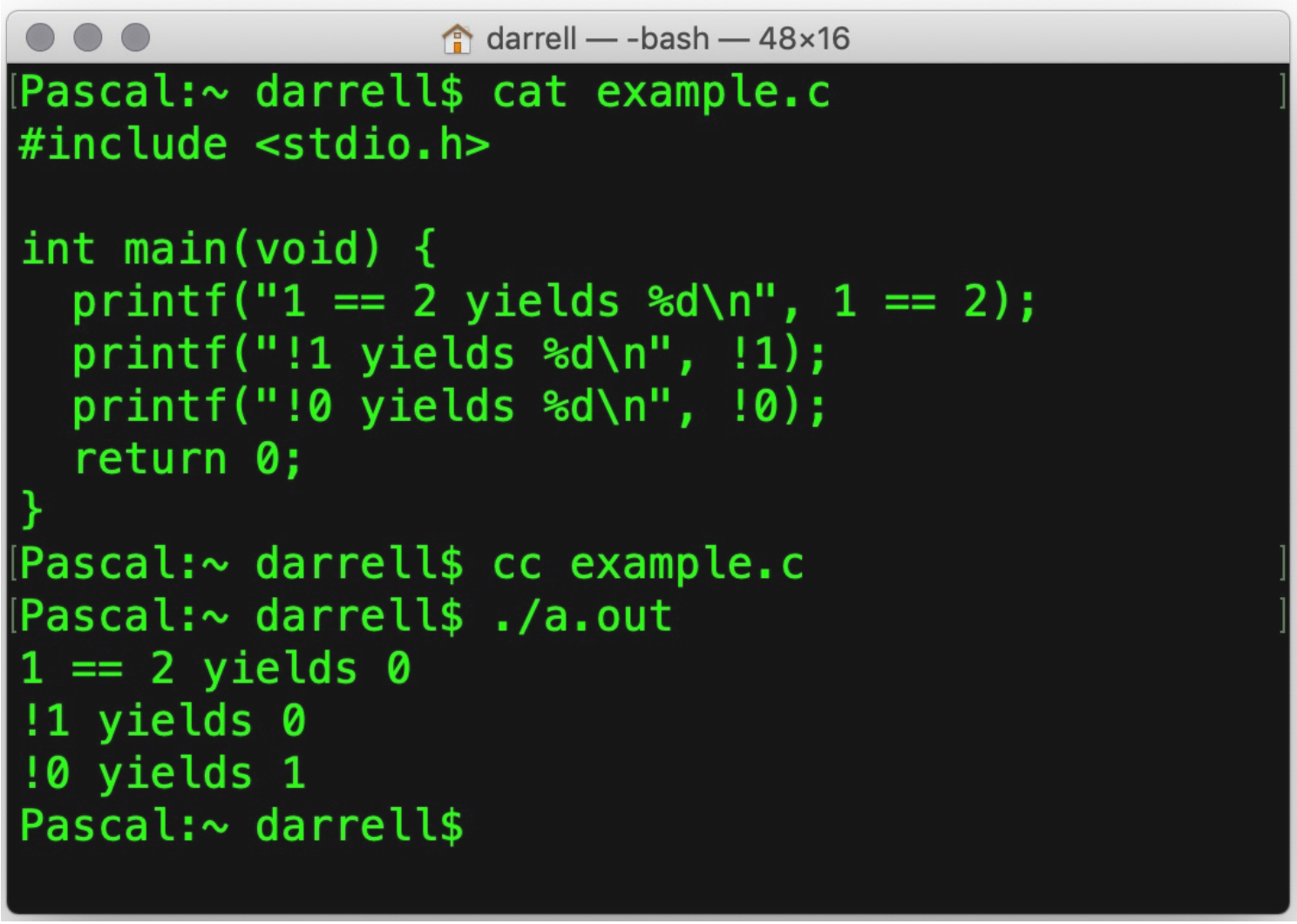
Comparison Operators

<	Less than
<=	Less than or equal
==	Equal
!=	Not equal
>	Greater than
>=	Greater than or equal

True and False in C

- In **C**, zero (`0`) is *false*.
- All that is not *false* is *true*.
- Logical expressions have type `int`.
- You can have `true` and `false` if you:

```
#include <stdbool.h>
```



C Operator	Mathematical Operation
`&&`	Boolean “and” (∧)
`||`	Boolean “or” (∨)
`!`	Boolean “not” (¬)

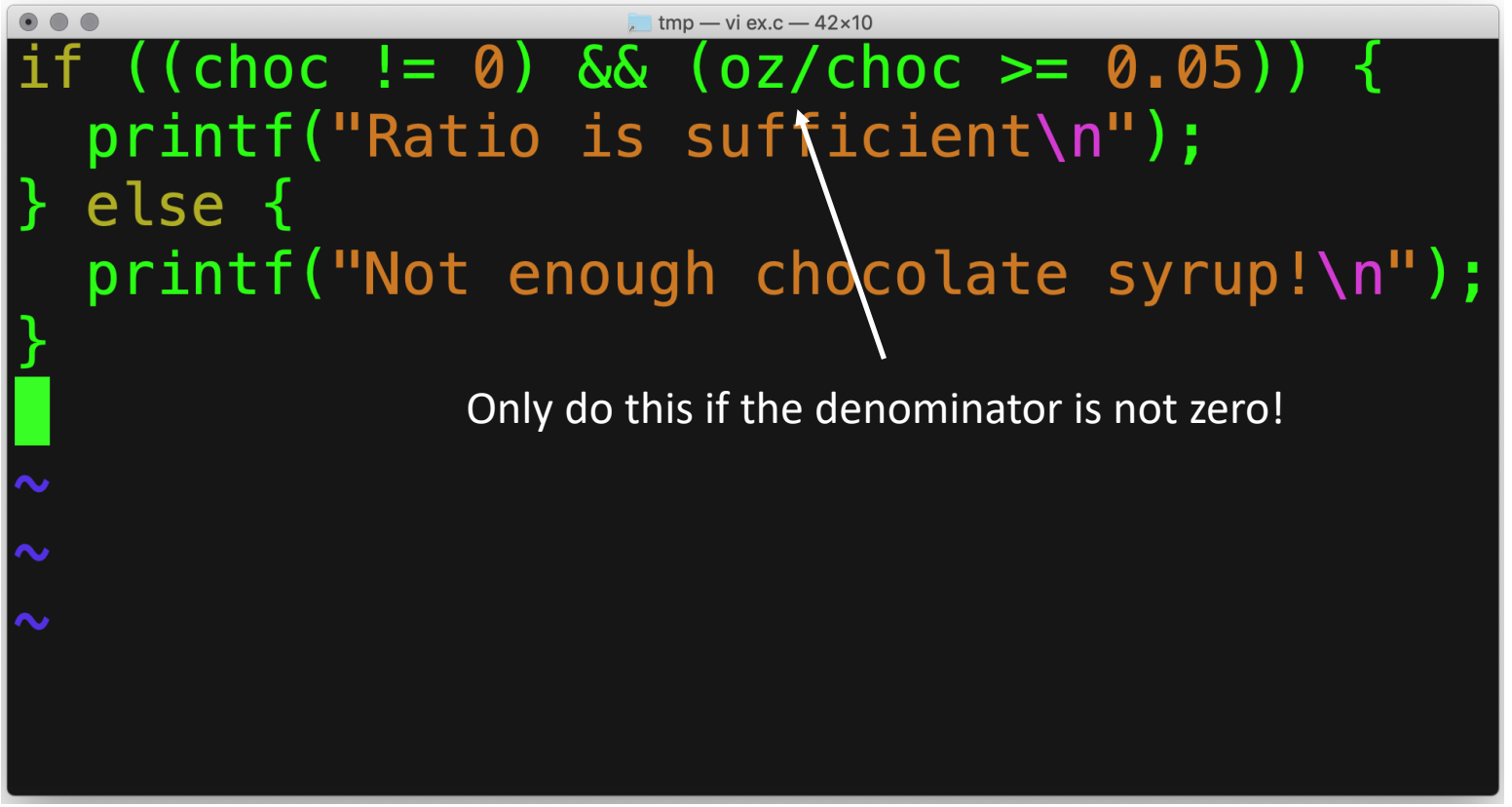
Short-circuit Evaluation

- *false* && anything is *false*
- *true* || anything is *true*

- Stop evaluating as soon as we know the result.

- Suppose we evaluated the entire expression:
  - We would be dividing by zero.

- We could follow a *null* pointer.
  - More on that later.



Infinite Loops

- All of these execute *forever*.
- The one you choose is a matter of *style*, not of substance.

- How do you ever escape?
  - Use the `break` statement.

```
while (1) {
  statement; ...
}

do {
  statement; ...
} while (1)

for (;;) {
  statement; ...
}
```

Let's Compute!

```
1.000000
1.500000
1.250000
1.375000
1.437500
1.406250
1.421875
1.414062
```

We'll compute $\sqrt{2}$.

But can't I just call a library routine?

No!

There is no magic — someone has to write the code.

The subfield of writing numerical programs is called *numerical analysis*.

$\sqrt{2}$ is that same as solving the equation:

$$x^2 - 2 = 0.$$

What else do we know?

We know $0 \leq x \leq 2$, so we'll start looking in the middle.

$$\sqrt{2} \approx 1.41421356237309504880168872420969807856967187537694807 3$$

```
long double Sqrt(long double x) {
  long double m, l = 0.0, h = (x < 1) ? 1 : x;
  steps = 0;
  do {
    steps += 1;
    m = (l + h) / 2.0;
    if (m * m < x) {
      l = m;
    } else {
      h = m;
    }
  } while (fabsl(l - h) > epsilon);
  return m;
}

int main(void) {
    printf("Sqrt(%5.2f) = %11.10Lf (δ = %11.10Lf, steps = %d)\n", 2.0, Sqrt(2.0),
           fabsl(sqrtl(2.0) - Sqrt(2.0)), steps);
  return 0;
}
```

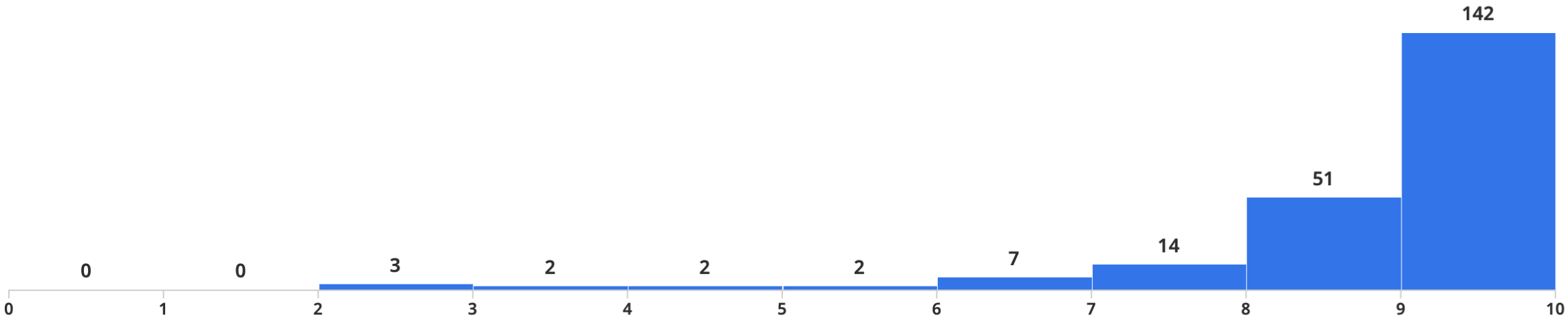
```
[eco :: Class/Examples/c » ./a.out
Sqrt(2.000000) = 1.4375000000 (δ = 0.023286437626905, steps = 5)
Sqrt(2.000000) = 1.4140625000 (δ = 0.000151062373095, steps = 8)
Sqrt(2.000000) = 1.4150390625 (δ = 0.000825500126905, steps = 11)
Sqrt(2.000000) = 1.4142456055 (δ = 0.000032043095655, steps = 15)
Sqrt(2.000000) = 1.4142074585 (δ = 0.000006103877001, steps = 18)
Sqrt(2.000000) = 1.4142141342 (δ = 0.000000571843213, steps = 21)
Sqrt(2.000000) = 1.4142135978 (δ = 0.000000035401410, steps = 25)
Sqrt(2.000000) = 1.4142135605 (δ = 0.000000001851493, steps = 28)
Sqrt(2.000000) = 1.4142135615 (δ = 0.000000000920170, steps = 31)
Sqrt(2.000000) = 1.4142135623 (δ = 0.000000000047055, steps = 35)
Sqrt(2.000000) = 1.4142135624 (δ = 0.000000000003877, steps = 38)
Sqrt(2.000000) = 1.4142135624 (δ = 0.000000000000671, steps = 41)
Sqrt(2.000000) = 1.4142135624 (δ = 0.000000000000046, steps = 45)
Sqrt(2.000000) = 1.4142135624 (δ = 0.000000000000004, steps = 48)
```

Quiz #1 Results

Review Grades for Quiz 1

Grades Not Published

All | Version A | Version B | Version C | Version D

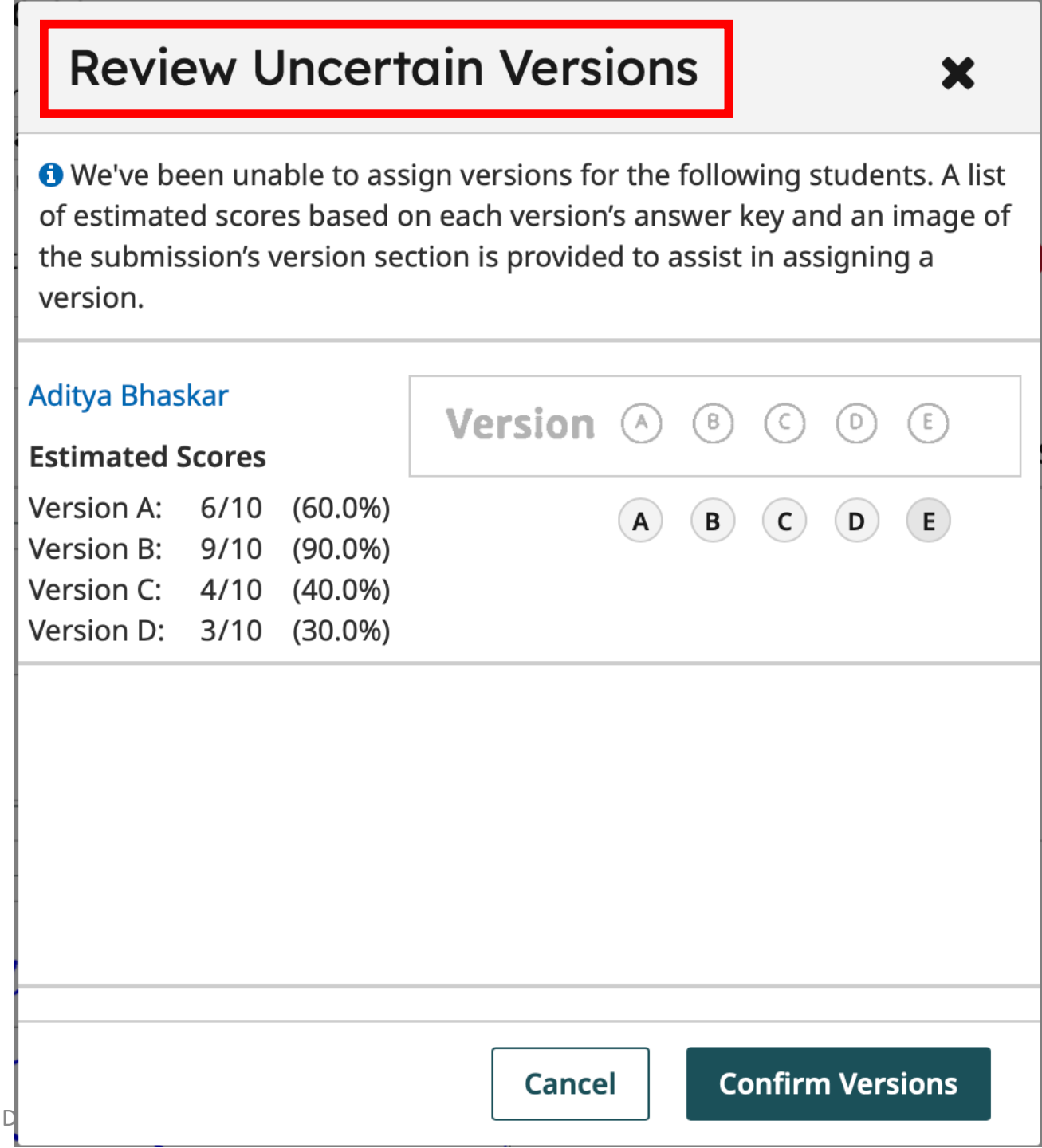


Minimum	Median	Maximum	Mean	Std Dev
2.0	9.0	10.0	8.81	--

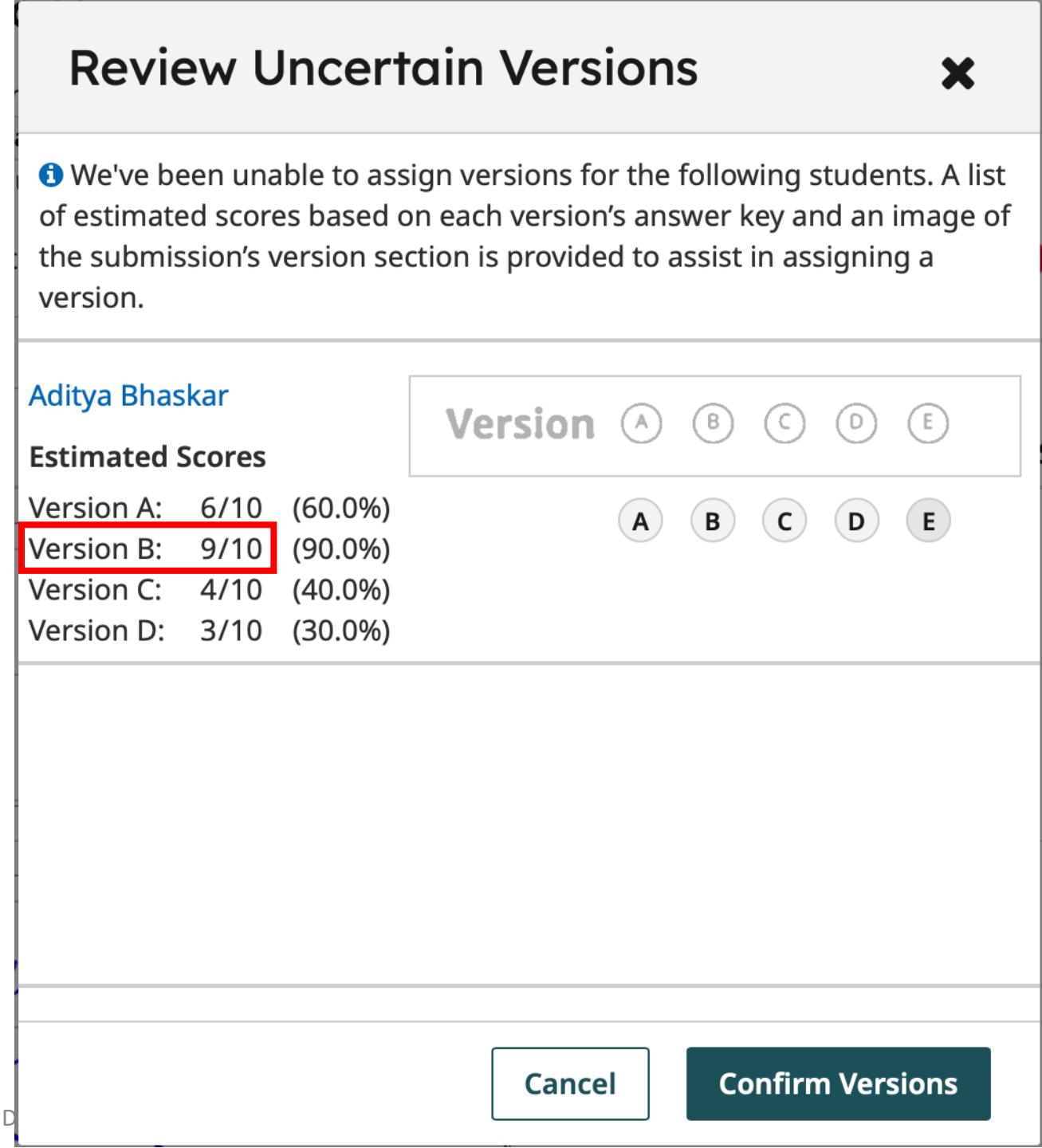
237 Students

Search

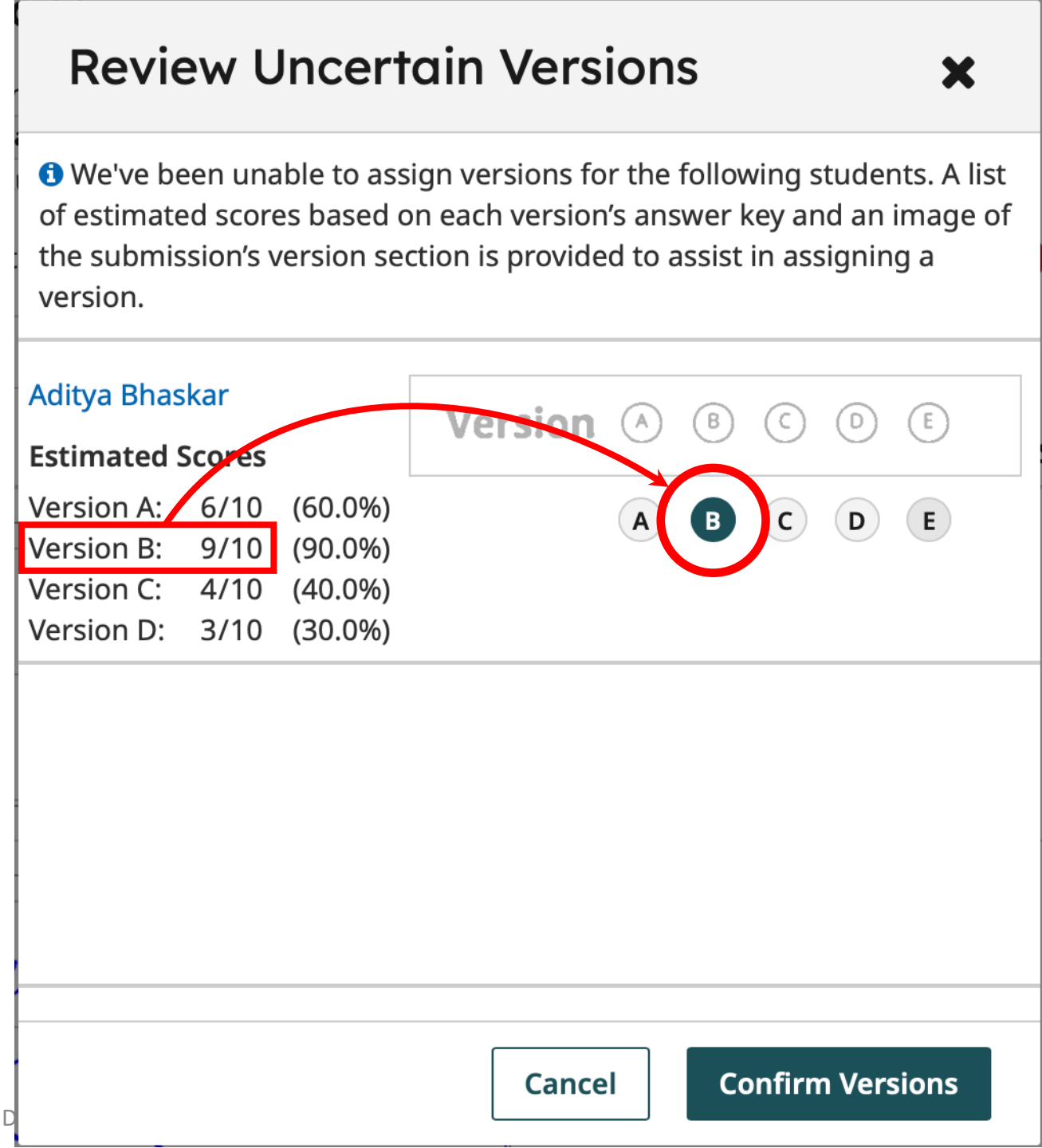
Quiz #1 Results



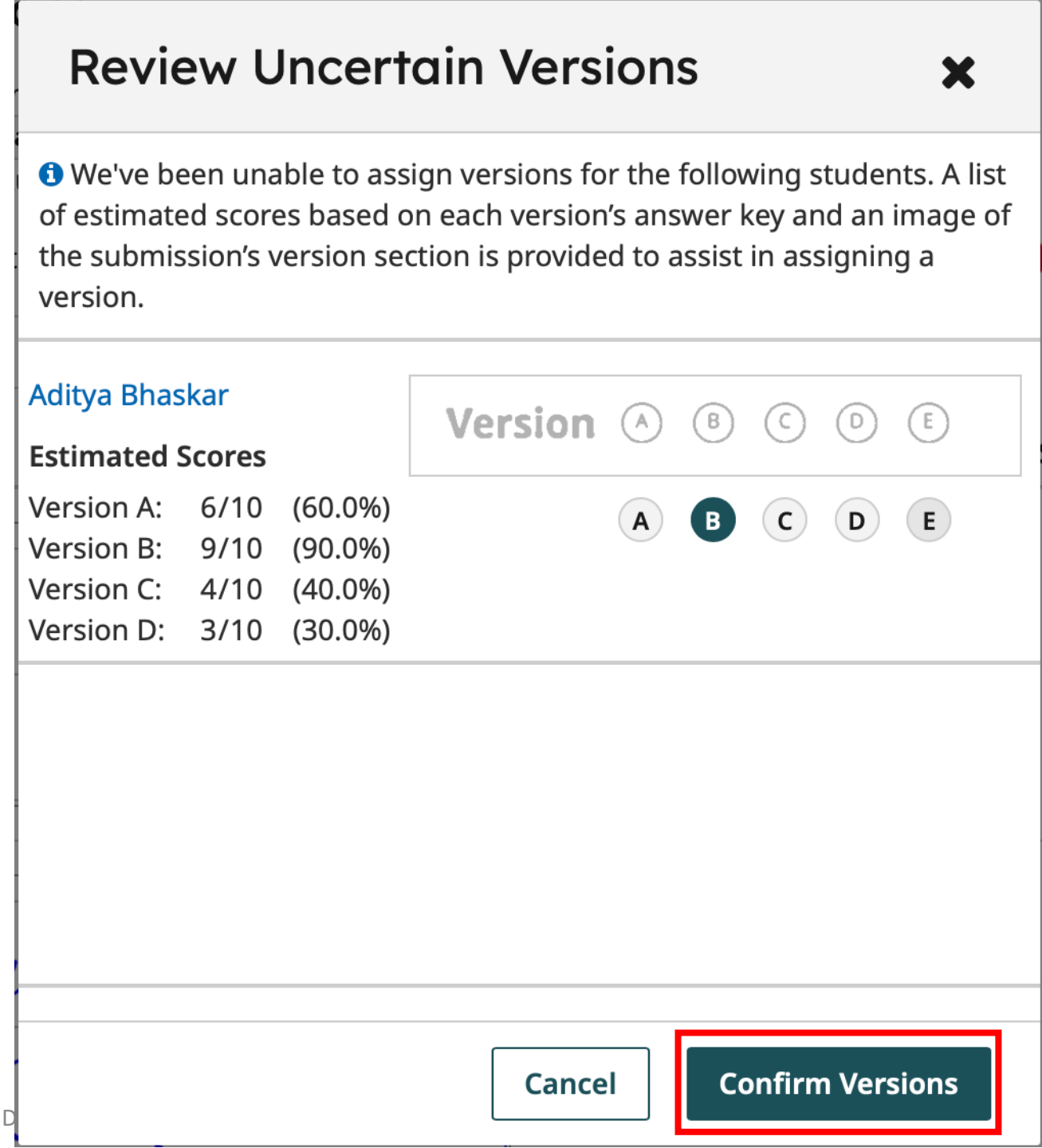
Quiz #1 Results



Quiz #1 Results

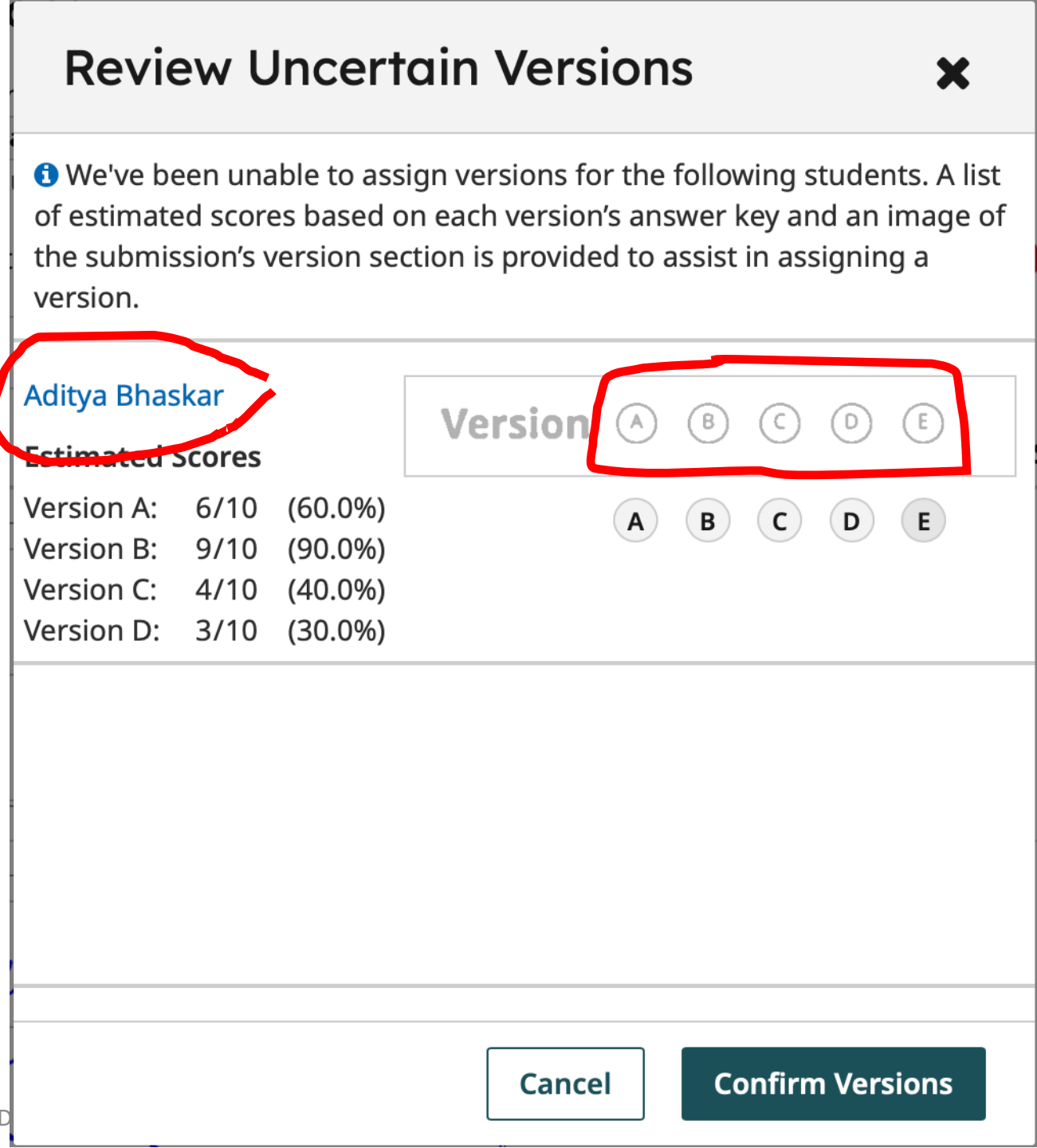


Quiz #1 Results



Uncertain Versions Award

- Winner: Aditya Bhaskar!



Tiny Writing Award

- Winner: Aaron Spalding