Weekly Programs in CS 1: Experiences with Many-Small Auto-Graded Programs

by Joe Michael Allen, Frank Vahid, Kelly Downey, and Alex Edgcomb

Dept. of Computer Science and Engineering

University of California, Riverside

This work was supported by the U.S. Dept. of Education (GAANN fellowship) and by Google

Problem

- College-level introductory programming courses (CS 1) have many well-known issues:
  - High student stress
  - Student dissatisfaction
  - Academic dishonesty
  - Low grades
  - High drop rates

- Weekly programming assignments form a large part of the student’s experience and are a key source of those issues.

Traditional 1-large Programming Assignments

- 1 Programming assignment a week
- Many concepts
- Confusion

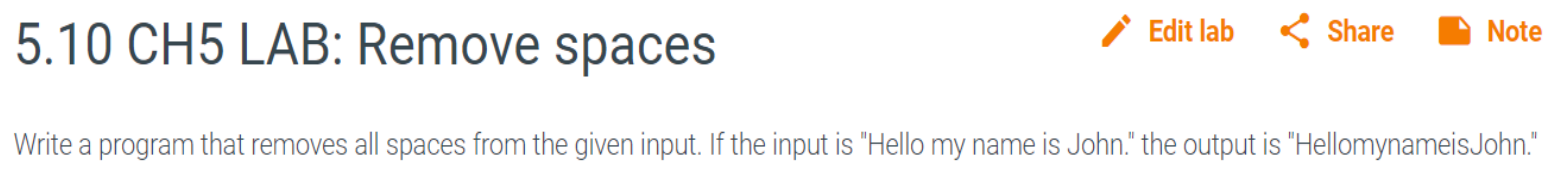
Traditional 1-large Programming Assignments Example

- 1) Modify the above program to ask the user to specify a number of tree trunk levels (“Enter trunk height: “), then use a loop to draw that many levels. Testing suggestion: If the user specifies 4 tree trunk levels, then the original tree should be drawn.
- (2) Modify the program again to ask the user to specify a number of tree trunk *’s per level ("Enter trunk width: “), then use a loop to draw that many *’s per level. You’ll need to use a nested loop in which the inner loop draws the *’s, and the outer loop iterates a number of times equal to the number of tree trunk levels.
- (3) Modify the program so that it only accepts odd numbers for the trunk width. Use a loop to continue prompting the user for a width until the number is odd.
- (4) Modify the program to ask the user to specify a number of tree leaves levels (“Enter leaves width: “), then use a nested loop to draw that many levels. You’ll need two inner loops for drawing the leaves: one for outputting spaces and one for outputting *’s. The outer loop iterates a number of times equal to the number of tree leaves levels. The top level of the leaves must have at least one *. You will have to modify this as well to only accept odd numbers for the number of leaves. Here is an example program execution. The user typed inputs have asterisks around them to clearly denote them as user inputs.
- Here is an example program execution when the user attempts to enter an even number for trunk width. A similar pattern should be followed, when a user attempts to enter an even number for leaves width as well. The user typed inputs have asterisks around them to clearly denote them as user inputs.

Our Solution

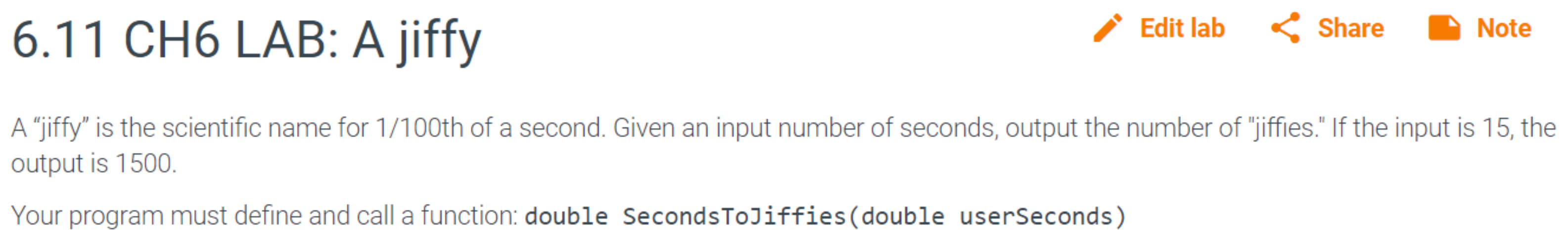
- Many Small Programs
  - 7 small labs per week
  - Specific concepts
  - 70% threshold
- Allowing collaboration

zyBooks



5.10 CH5 LAB: Remove spaces

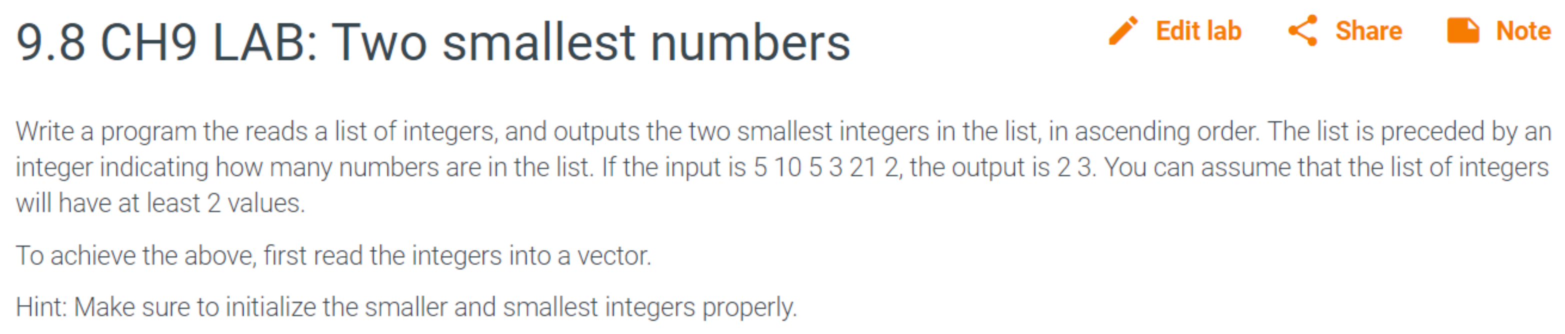
Write a program that removes all spaces from the given input. If the input is "Hello my name is John." the output is "HellomynameisJohn."



6.11 CH6 LAB: A jiffy

A "jiffy" is the scientific name for 1/100th of a second. Given an input number of seconds, output the number of "jiffies." If the input is 15, the output is 1500.

Your program must define and call a function: `double SecondsToJiffies(double userSeconds)`



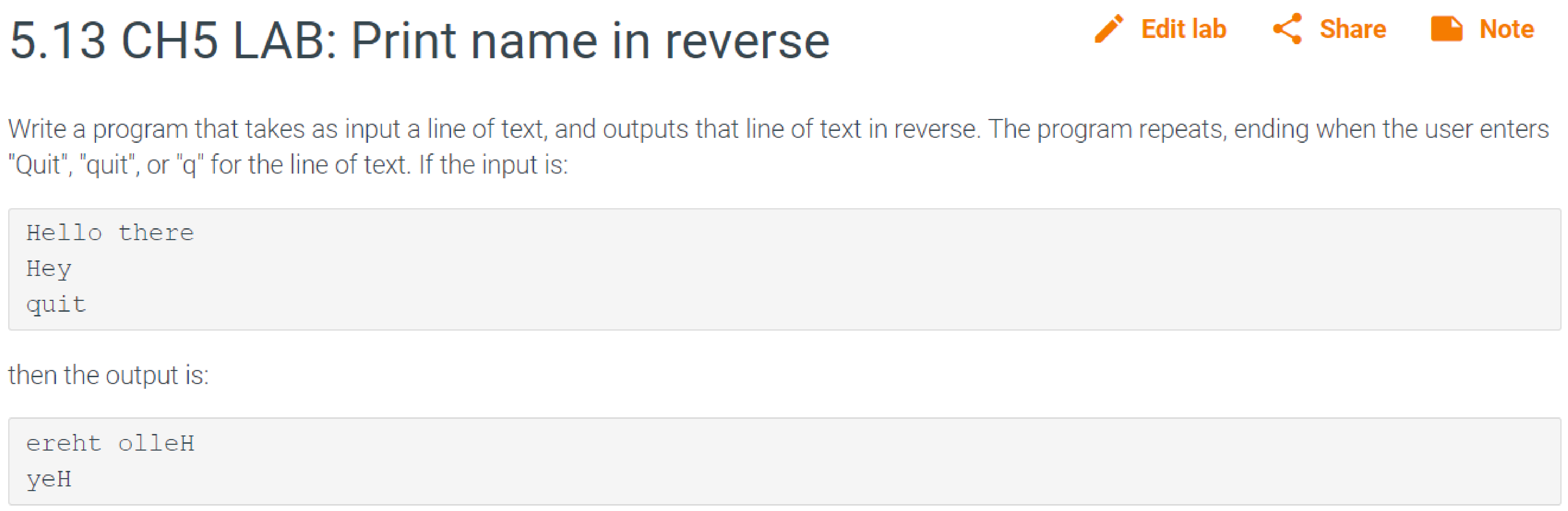
9.8 CH9 LAB: Two smallest numbers

Write a program the reads a list of integers, and outputs the two smallest integers in the list, in ascending order. The list is preceded by an integer indicating how many numbers are in the list. If the input is 5 10 5 3 21 2, the output is 2 3. You can assume that the list of integers will have at least 2 values.

To achieve the above, first read the integers into a vector.

Hint: Make sure to initialize the smaller and smallest integers properly.

Many Small Programs - Prompt



5.13 CH5 LAB: Print name in reverse

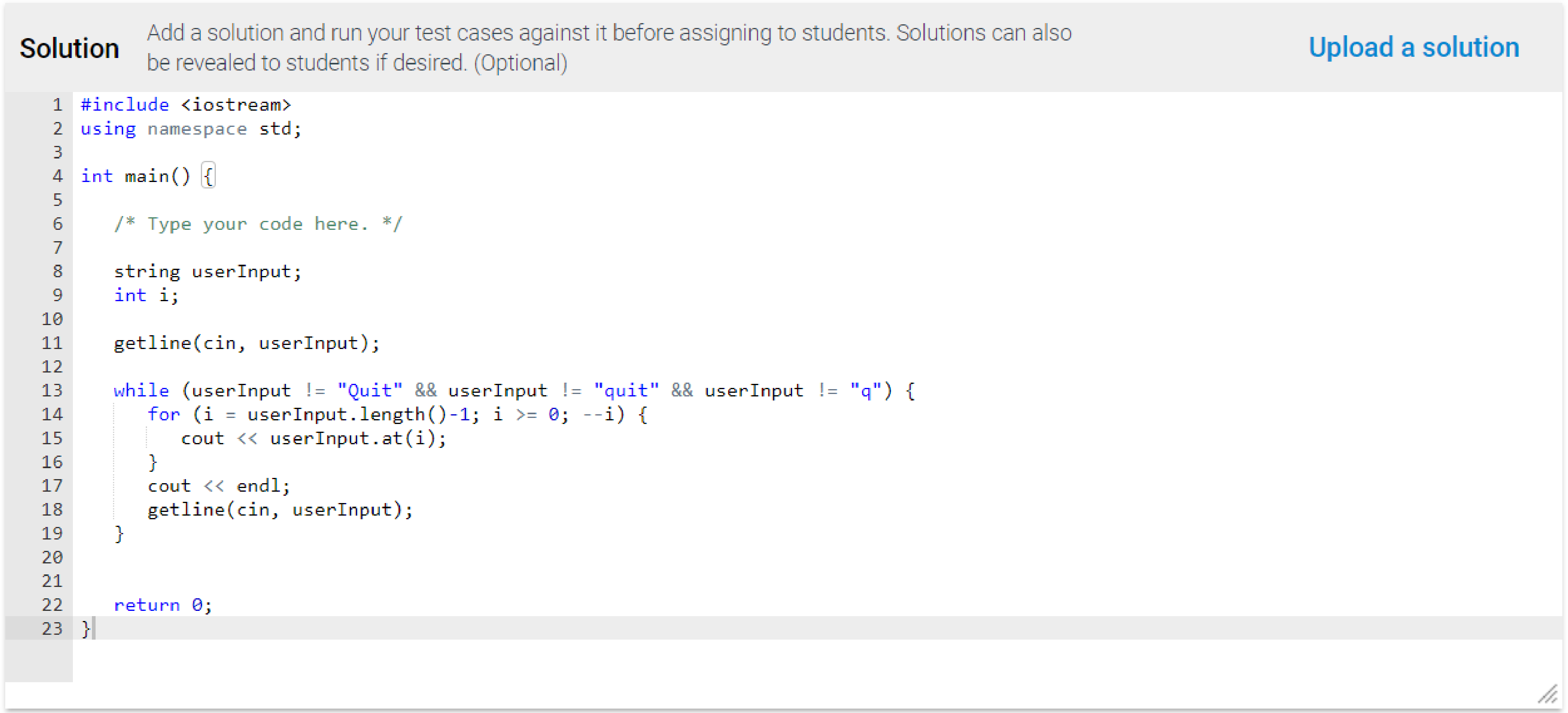
Write a program that takes as input a line of text, and outputs that line of text in reverse. The program repeats, ending when the user enters "Quit", "quit", or "q" for the line of text. If the input is:

```
Hello there
Hey
quit
```

then the output is:

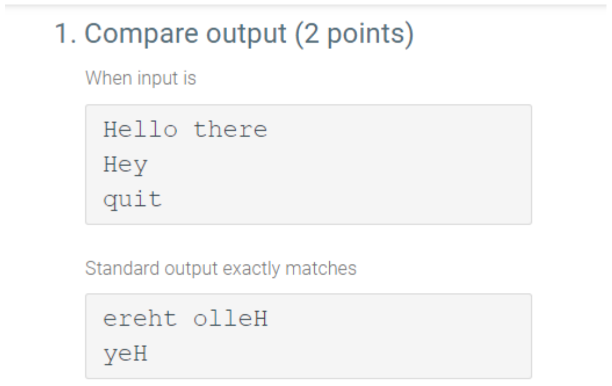
```
ereht olleH
yeH
```

Many Small Programs - Solution


Solution
Add a solution and run your test cases against it before assigning to students. Solutions can also be revealed to students if desired. (Optional)
Upload a solution

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

int main() {

   /* Type your code here. */

   string userInput;
   int i;

   getline(cin, userInput);

   while (userInput != "Quit" && userInput != "quit" && userInput != "q") {
      for (i = userInput.length()-1; i >= 0; --i) {
         cout << userInput.at(i);
      }
      cout << endl;
      getline(cin, userInput);
   }


   return 0;
}
```

Many Small Programs – Test Cases



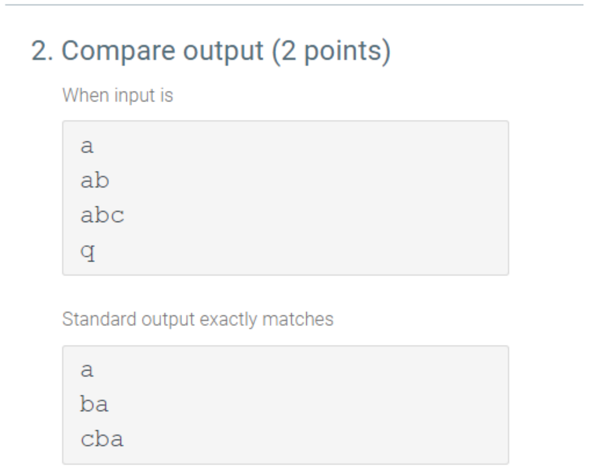
1. Compare output (2 points)

When input is

```
Hello there
Hey
quit
```

Standard output exactly matches

```
ereht olleH
yeH
```



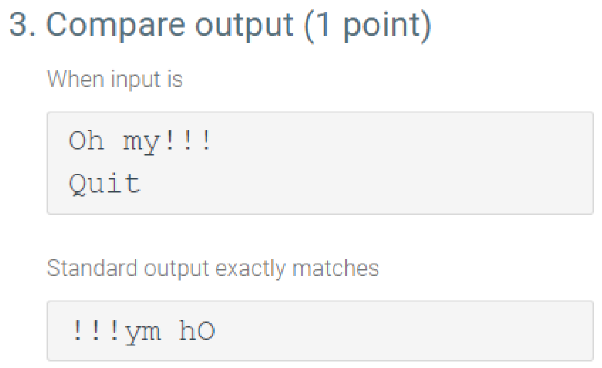
2. Compare output (2 points)

When input is

```
a
ab
abc
q
```

Standard output exactly matches

```
a
ba
cba
```



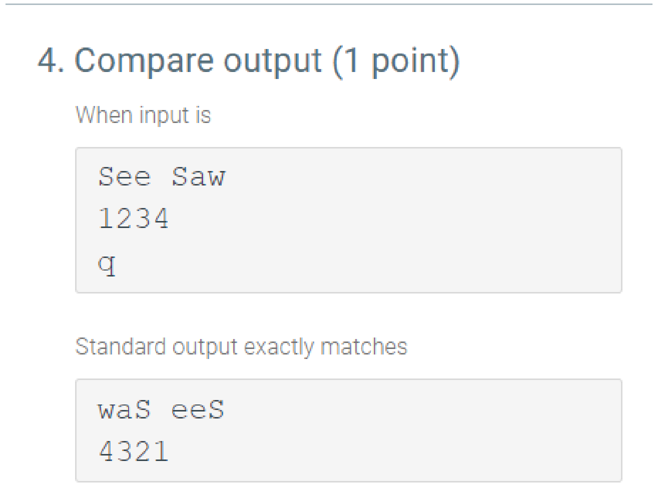
3. Compare output (1 point)

When input is

```
Oh my!!!
Quit
```

Standard output exactly matches

```
!!!ym hO
```



4. Compare output (1 point)

When input is

```
See Saw
1234
q
```

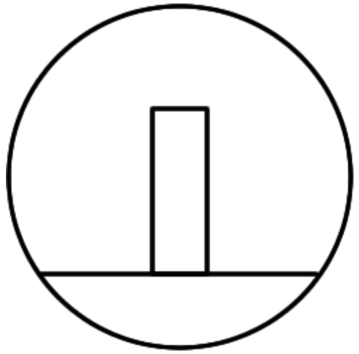
Standard output exactly matches

```
waS eeS
4321
```

Methodology

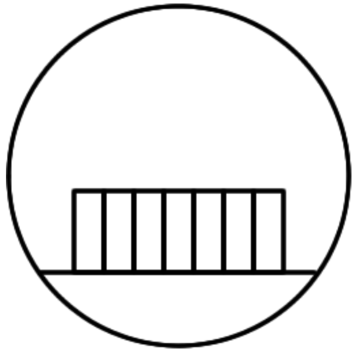
- Study conducted on a CS10 course at UCR during Spring 2017

Control Group



2 in-person sections (~160 students)
One-large program a week
No collaboration
Program assignments: 25%, Midterm: 20%

Experimental Group



1 online section (~80 students)
Many Small Programs
Collaboration allowed
Program assignments: 15%, Midterm: 30%

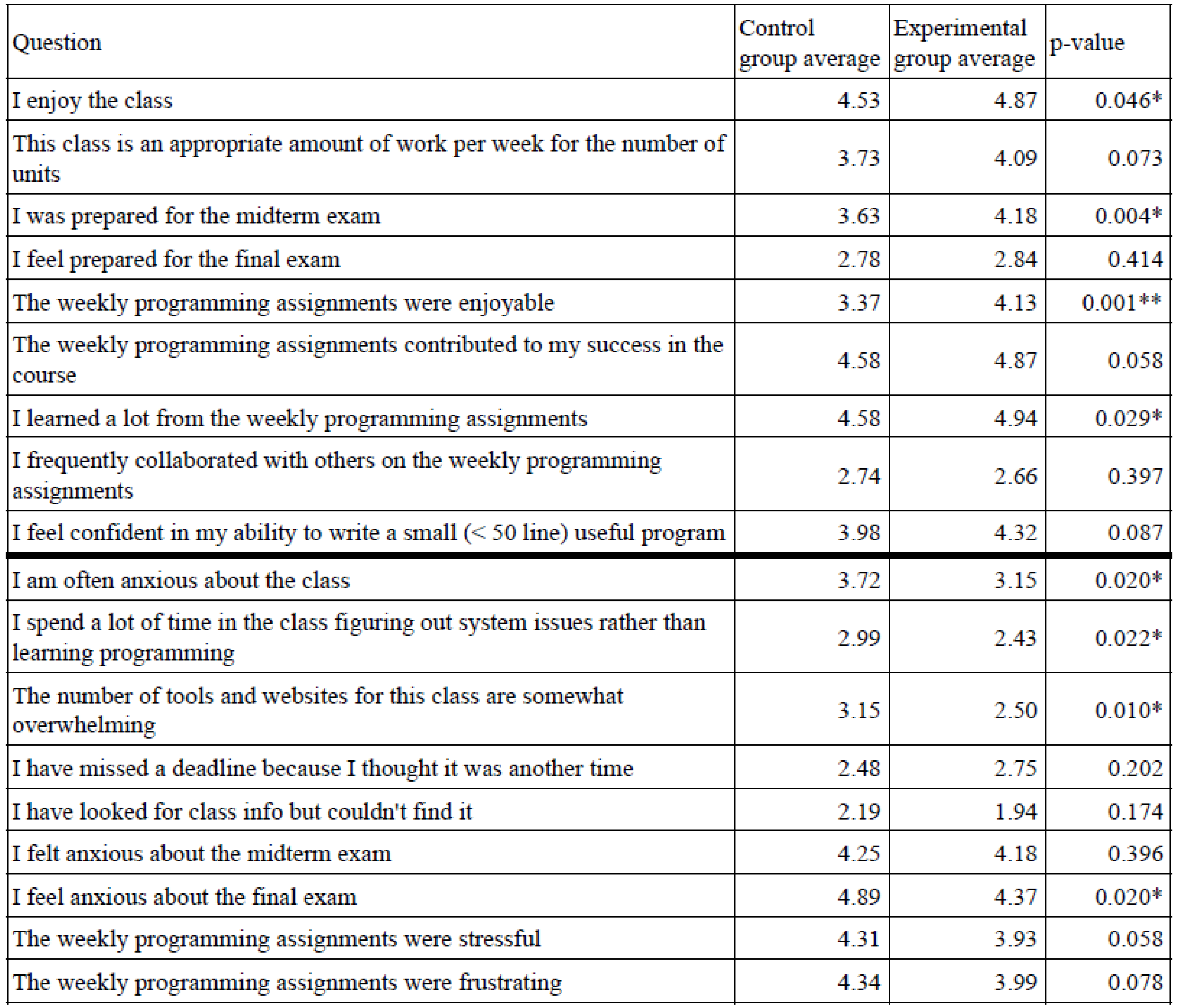
Methodology cont.

- Student Surveys (“Stress Survey”)
  - Asked students about their experience
  - Survey given during week 8 of the quarter

- Grade Performance
  - Averaged grade percentages for Final, Midterm, Reading Activities, Challenge Activities, and Programming Activities

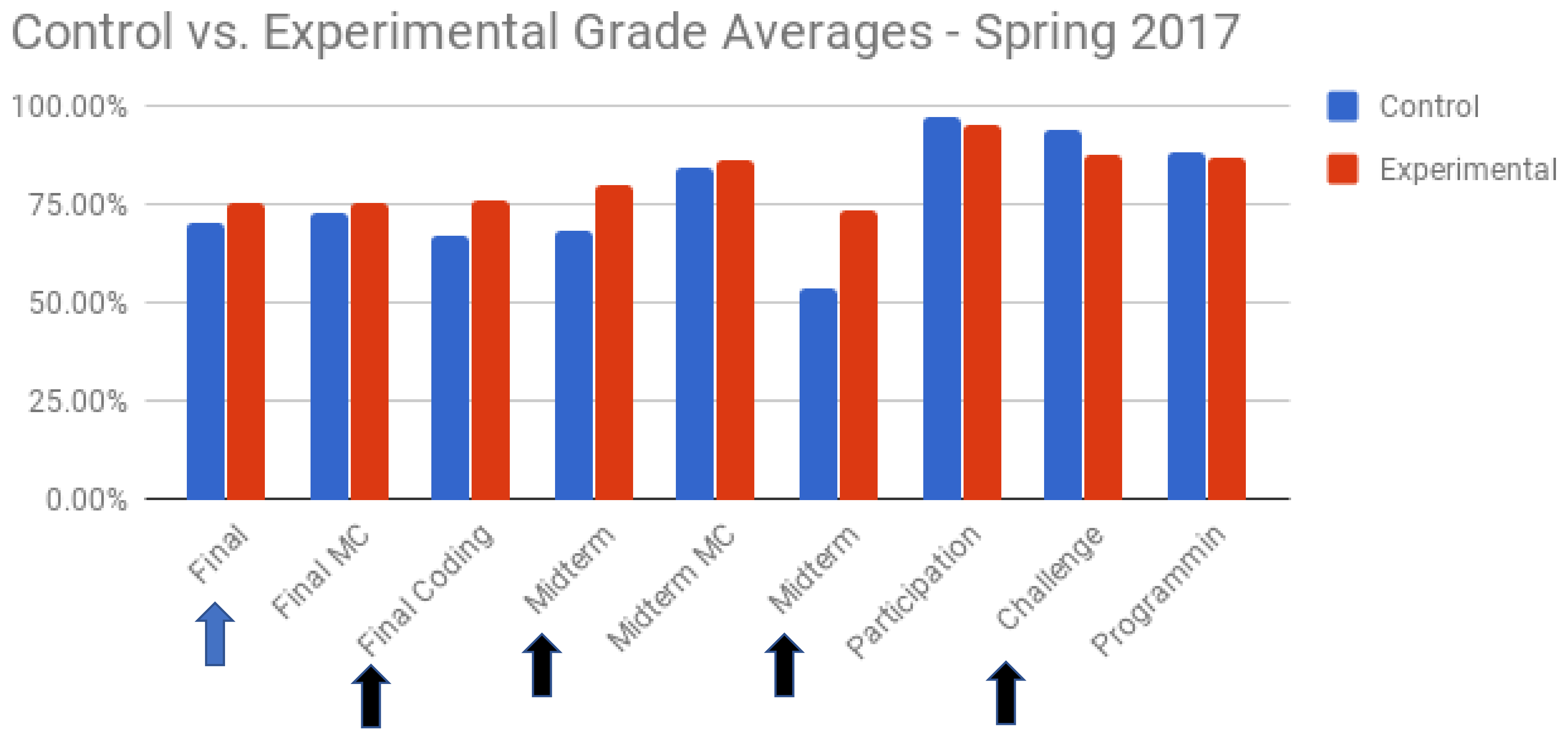
Results – Stress Survey

- Experimental group preferred the class 9% more



Question	Control group average	Experimental group average	p-value
I enjoy the class	4.53	4.87	0.046*
This class is an appropriate amount of work per week for the number of units	3.73	4.09	0.073
I was prepared for the midterm exam	3.63	4.18	0.004*
I feel prepared for the final exam	2.78	2.84	0.414
The weekly programming assignments were enjoyable	3.37	4.13	0.001**
The weekly programming assignments contributed to my success in the course	4.58	4.87	0.058
I learned a lot from the weekly programming assignments	4.58	4.94	0.029*
I frequently collaborated with others on the weekly programming assignments	2.74	2.66	0.397
I feel confident in my ability to write a small (< 50 line) useful program	3.98	4.32	0.087
I am often anxious about the class	3.72	3.15	0.020*
I spend a lot of time in the class figuring out system issues rather than learning programming	2.99	2.43	0.022*
The number of tools and websites for this class are somewhat overwhelming	3.15	2.50	0.010*
I have missed a deadline because I thought it was another time	2.48	2.75	0.202
I have looked for class info but couldn't find it	2.19	1.94	0.174
I felt anxious about the midterm exam	4.25	4.18	0.396
I feel anxious about the final exam	4.89	4.37	0.020*
The weekly programming assignments were stressful	4.31	3.93	0.058
The weekly programming assignments were frustrating	4.34	3.99	0.078

Results - Grades



Conclusion

- Students that had MSPs prefer the course 9% more than students that had the traditional programming assignments.
- Students that had MSPs outperformed students that had traditional programming assignments on the Final and the Midterm, and did about the same on the other class categories.

Future Work

- After 1 year, are we still seeing similar results with student perception and grades?
- If we allow collaboration, do students actually collaborate – if so, how much?
- Does having MSPs in CS1 affect student performance in CS2?
- Do students start programming assignments earlier if they are MSP or larger assignments?
- Do students perform better if they have all MSPs or mostly MSPs with some large assignments?