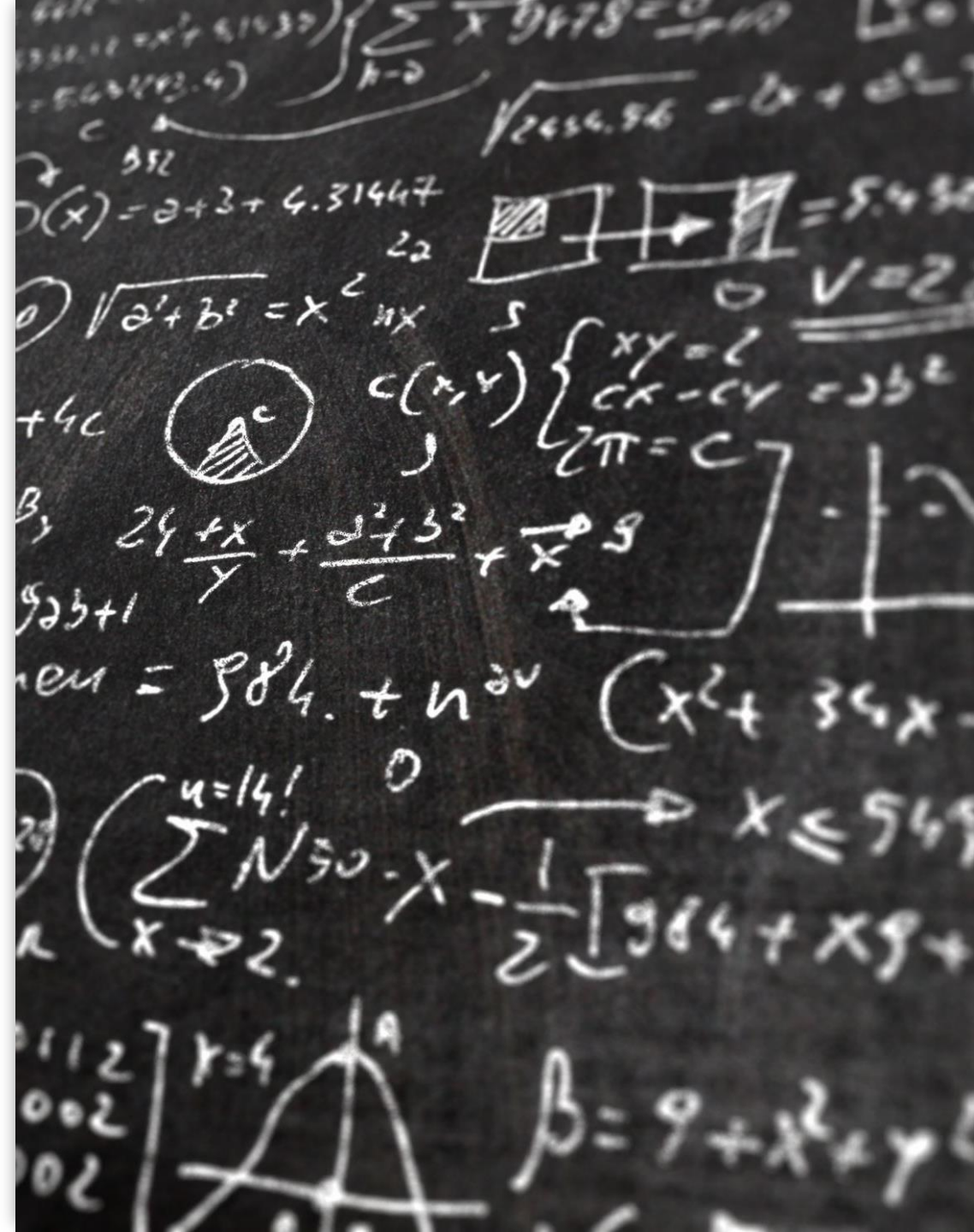


SCC.111 Software Development – Lecture 13: Quiz Solutions!

Adrian Friday, Nigel Davies, Hansi Hettiarachchi, Saad Ezzini

This lecture

- Examining common mistakes on the quiz
- Working through our approaches to this

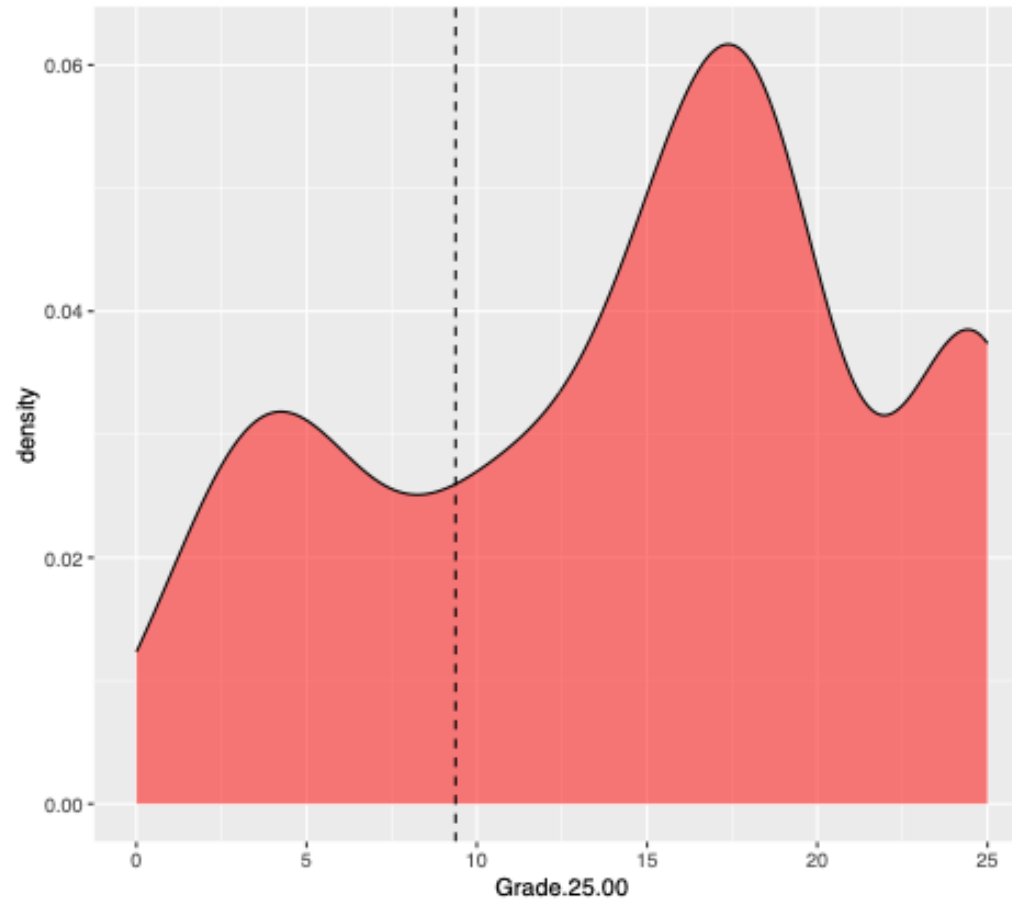


Why did we do a quiz?

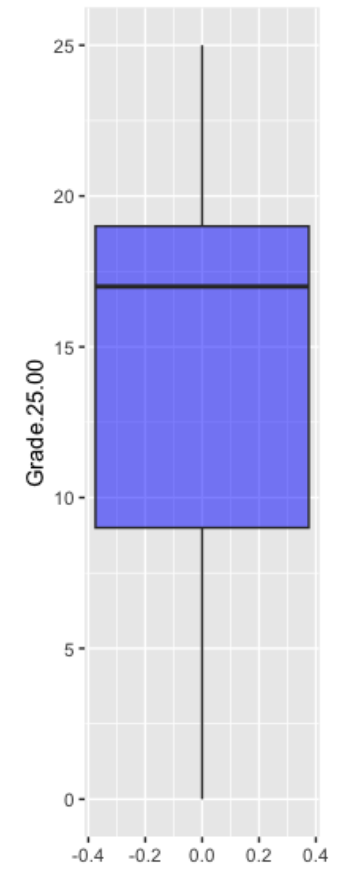
- To check how you're getting on
- So you can check how you're getting on!



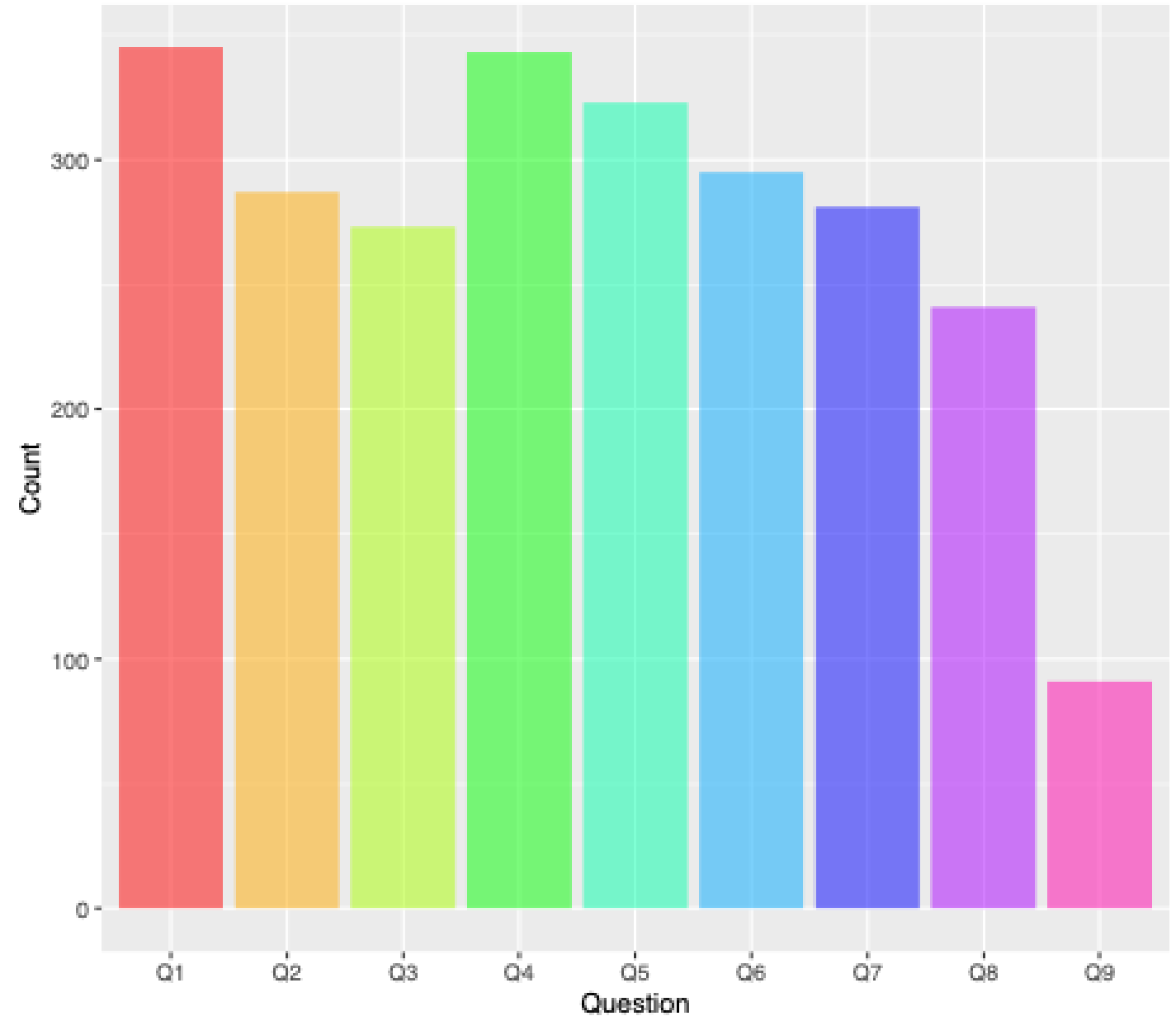
Overall,
people did
well!!!




Median score = 68%!



But generally
coding
questions
were harder!



A blurred office scene with a laptop, a mug, and papers on a desk. The text "Let's look at where some went wrong!" is overlaid on the left side of the image.

Let's look at where
some went wrong!

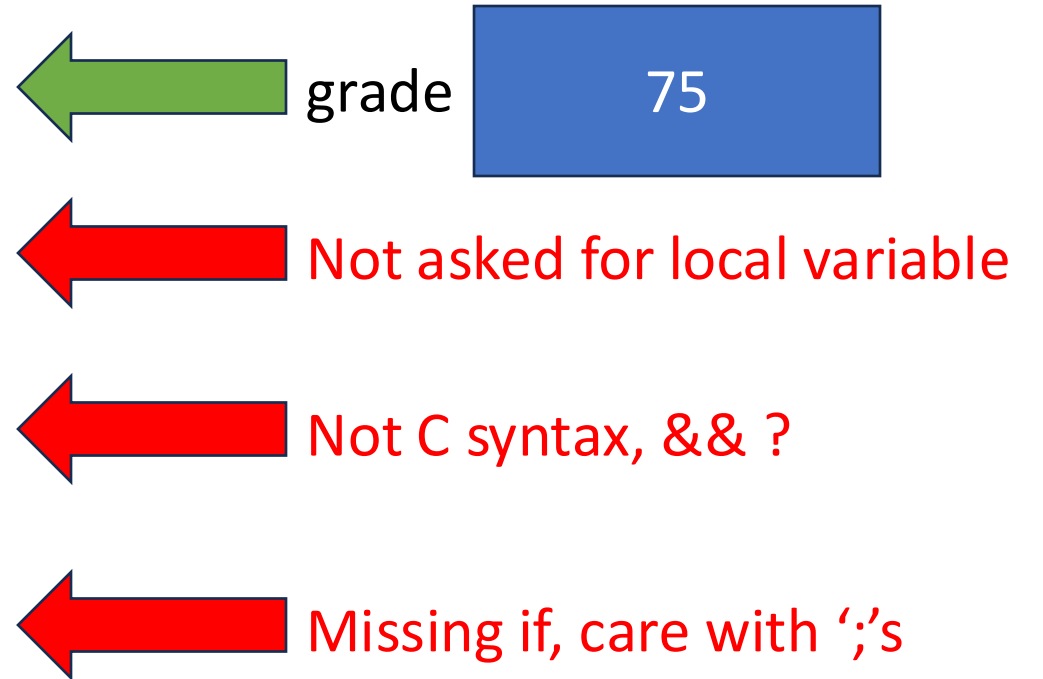
"If" questions

- Given the following pair of printf() statements as your starting point. Write a selection statement that obeys the following rules:
 - **True** is printed when grade is between 70 and 100 inclusive
 - **False** is printed otherwise

```
void first_class(int grade)
{
    float x;
    if ((x>70) (x<100))
        printf("True\n");
    else ((x<70) (x>100));
        printf("False\n");
}
```




```
void first_class(int grade)
{
    float x;
    if ((x>70) (x<100))
        printf("True\n");
    else ((x<70) (x>100));
        printf("False\n");
}
```



Let's fix it!

“Loop” questions

- Given the following `printf()` statement as your starting point. Write a loop in C which counts from 1 to n in increments of 2.

```
void looper(int n)
{
    int iterator;
    while (iterator < n) {
        iterator + 2;
        printf("Loop iteration %d\n", iterator);
    }
}
```



```
void looper(int n)
```

```
{
```

```
    int iterator;
```

```
    while (iterator < n) {
```

```
        iterator + 2;
```

```
        printf("Loop iteration %d\n", iterator);
```

```
    }
```

```
}
```



Uninitialised local variable



Does it execute?



No left hand side assignment

Let's fix it!

“Total” question

- Given the following template, write a snippet of C that adds up the numbers in the array and returns the correct total

```
int calc_total(int numbers[], int arrayLength)
{
    // note: numbers is an integer array of arrayLength elements
}
```

Close, but...

```
int main()
{
    int u[5];
    int i, sum = 0;
    for(int i=0; i<5, i++)
        scanf("%d", &u[i])

    printf("total\n");
    for(int i=0; i<5, i++)
        sum = sum + u[i]

    printf("%d", sum);
    return 0;
}
```



Let's work this through step by step...

```
int calc_total(int numbers[], int arrayLength)
{
}
```

Result 'returned' for this one

arrayLength	3
numbers	1
	2
	3

The solution is actually relatively simple...

```
int calc_total(int numbers[], int arrayLength)
{
    int total = 0;

    for (int i = 0; i < arrayLength; i++)
        total += numbers[i];

    return total;
}
```

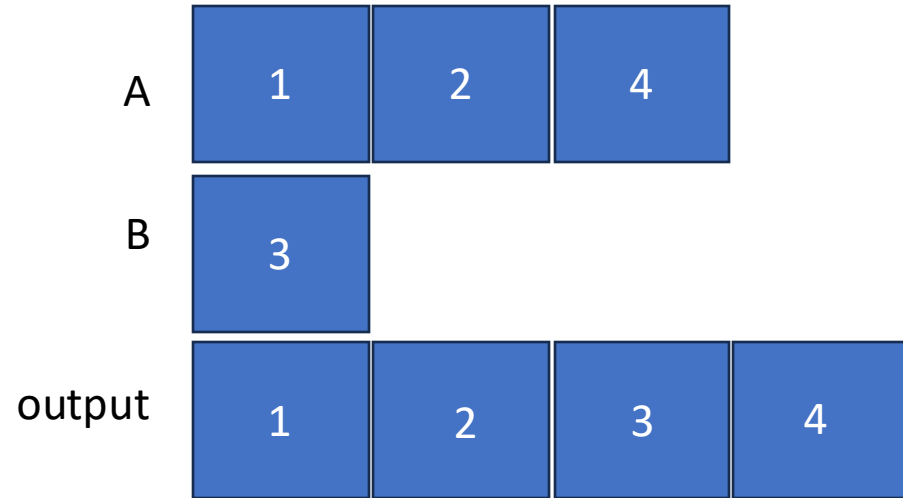

The 'sting' in the tail

- Implement the body of the function 'merge_sort'. You can assume the functions work in terms of whole decimal numbers, and that the input arrays will *not exceed* 10 elements in length (the output could be up to 20 elements long).
 - Merge sort works as follows.
 - Input A: 1,2,4
Input B: 3
Final output: 1,2,3,4

Algorithm:

1. Given two sorted input lists, copy the smallest from either list to the output
2. Repeat this process until all data is processed
3. Note that if either list is exhausted, you can just copy the remainder of the remaining list to the output

Let's work through an approach to this on the Visualiser



Is 1 less than 3?

Yes, copy to output

Is 2 less than 3?

Yes, copy to output

Is 4 less than 3?

No, copy 3 to output

B's done, copy rest of A



Summary

- Worked through some questions that were problematic
- Discussed some syntax and logical errors
- Remember to read the spec carefully!
- *Next lecture: even more powerful uses of pointers!*