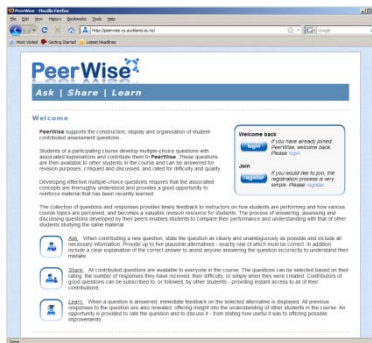


# PeerWise

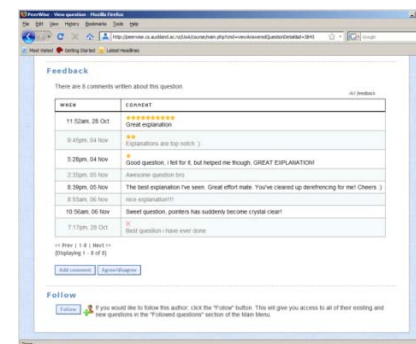
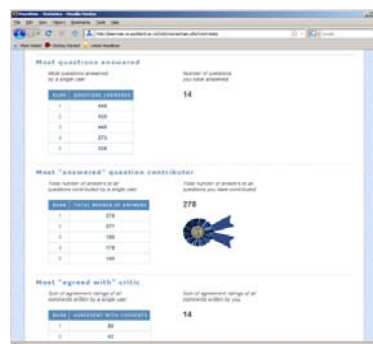
## Students sharing and evaluating their MCQs

<http://peerwise.cs.auckland.ac.nz>



PeerWise is a web-based MCQ repository, that allows students to:

- develop new questions with associated explanations
- answer existing questions and rate them for quality and difficulty
- take part in discussions
- subscribe to authors they like
- compete with other students to appear on leaderboards



Paul Denny  
The University of Auckland

# Motivations

## Encourage student reflection

“Writing my multi-choice question was hard given that I **had to think of the possible wrong solutions** students would fall for and **required a lot of thinking** from me, which in the end was a lot of help because I was just about **able to answer any question that was on the same topic** as my question”

## Provide opportunities for self-assessment and peer comparison

“Being able to see how other people answered was great as it allowed me to **recognise at which level I was at compared to everyone else**”

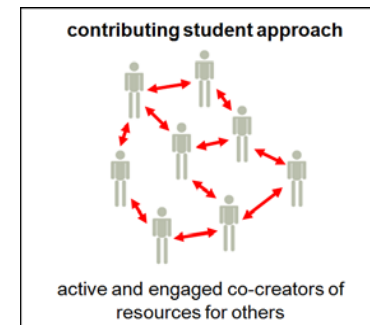
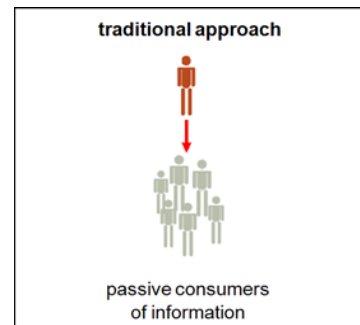
## Leverage intellectual capacity of the class

Virtually no instructor moderation required  
UoA 2007, ENGGEN 131, 6 weeks:

- 570 students
- 1,700 questions
- 35,000 answers

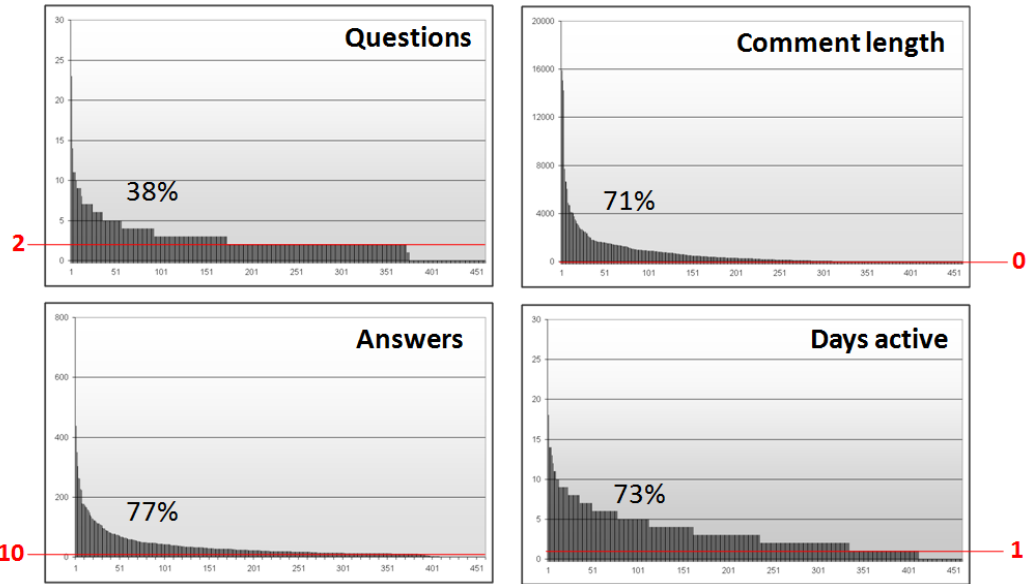
“You have the potential to do as many questions as you can possibly dream of doing. You often **come across questions which have arisen from problems others have come across** and solved - many of which I have not experienced myself”

## Support a “contributing student” pedagogy

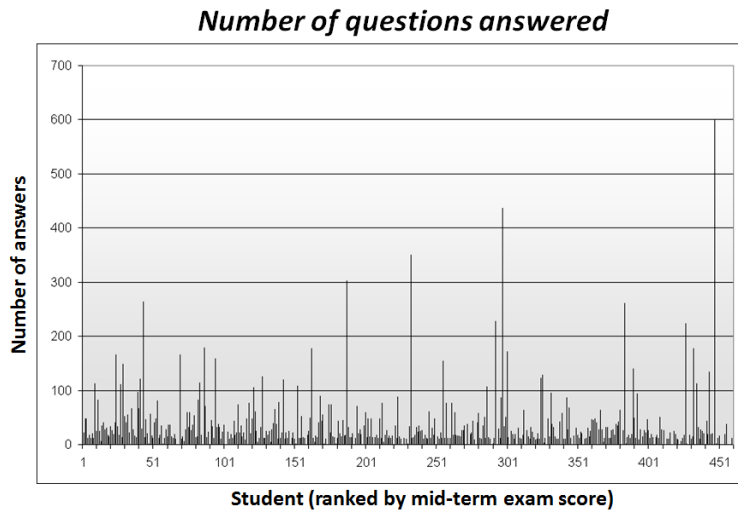


# Usage patterns

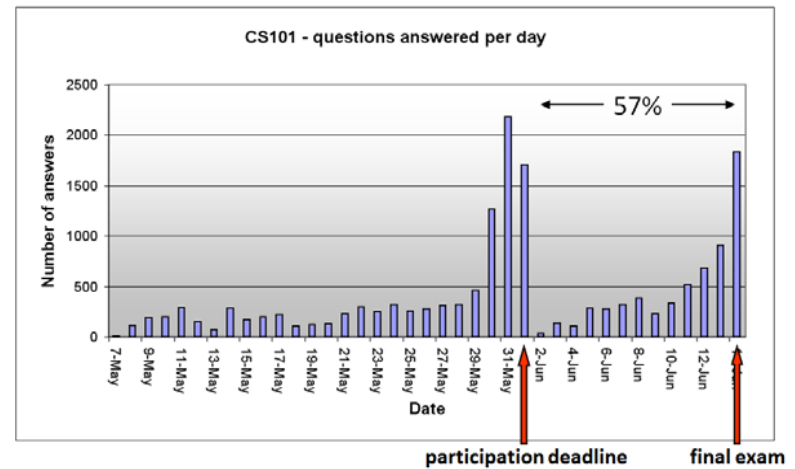
Many students contribute more than the minimum requirements:



Participation from all ability levels:



Voluntary exam study use:



# Question quality

## Question

Which of the following loops *could* you use to cycle through **all** elements of the following array **without** going out of bounds and causing the program to crash?

```
int[] array = new int[15];
```

Assume that in all cases, **array[i]** will be used without any other math operating inside the square brackets (dereferencing operator).

## Explanation

When dealing with arrays, there are a few things to remember. 1) When created, the value used inside the square brackets indicates the length of the array, or how many elements it can contain. The length counts from a starting point of 1. The INDEX however, begins at 0. Meaning that in this case, where we created our integer array with a length of 15, the valid index values are 0-14.

(C) is the correct answer because:

$i = \text{array.length} - 1$ , evaluates to 14. The last index of the array.

The conditional statement will go down to AND include 0, the first index of the array, but will not pass this point and go out of bounds.

$i--$  means subtract 1 from  $i$  every time it goes around, so every number from 14 to 0 will be a value of  $i$  during the loops lifespan.

Why are the other's incorrect?

(A) This loop would crash at the end.

$i = 0$ , this is fine, it is the first value of the index and is correct.

BUT

The conditional inside the while loop is:  $i \leq \text{array.length}$ , which means it can be less than OR equal to  $\text{array.length}$ , which is 15. The last index is 14, thus when it attempted to find index 15 of the array, it would crash with an out of bounds error.

(B) This suffers the exact same problem as A, but has been rendered in 'for' loop format.

(D) The loop shown for D would not crash, but nor would it completely cycle through all values of this array.

$\text{int } i = \text{array.length} - 1$  as discussed above will result in 14 which is correct for the last index of our array,

However,

The conditional:  $i > 0$  will not ever allow this loop to check index 0. It will stop after cycling through 1.

(E) This loop again will not crash, but will not cycle completely through all values of this array.

$\text{int } i = 1$  means that 0 will not be evaluated.

the conditional inside the while loop will stop the cycle correctly at 14 to prevent the crash.

$i++$  means that it will increment the index until the conditional stops this loop.

## Alternatives

Alt A 5 (4.63%)	<pre>int i = 0; while (i &lt;= array.length) { i++; // code }</pre>
Alt B 22 (20.37%)	<pre>for (int i = 0; i &lt;= array.length; i++) { // code }</pre>
Alt C 46 (42.59%)	<pre>for (int i = array.length - 1; i &gt;= 0; i--) { // code }</pre>
Alt D 17 (15.74%)	<pre>for (int i = array.length - 1; i &gt; 0; i--) { // code }</pre>
Alt E 18 (16.67%)	<pre>int i = 1; while (i &lt; array.length) { i++ // code }</pre>

## Comments

★★★★★★★★

Sneaky. Very good, although it is not how one normally thinks of looping through an array, it is a common pitfall and very well highlighted. Well explained as well.

★★★★

Good testing of understanding of loops. Awesome.

★★

while I think the question is quite confusing, this is a great question.(and very great explanation by the way).

★★

Nice question. A way of looping I hadn't considered until now, but still applicable and within the scope of the course.

★★

Thinking about the various different increments and conditions which can be used in a loop! I think it's a nice change from the usual loop questions that normally involve an ascending value of  $i$ . Brilliant:)

★

Good questions to understand loops and array... Good explanations as well... thank you...

Good questions...

### Question





What is the appropriate boolean variables need to be stored in A and B if the following returns false:

```
!A || B && !B || A
```

### Explanation

The Answer is A:  
When A = False, B = True;  
`!A || B && !B || A` =>  
`(True) || (True) && (False) || (False)`  
=> `True && False = False`

### Alternatives

Alt A  2 (14.29%)	A = False B = True
Alt B  3 (21.43%)	A = True B = True
Alt C  0 (0.00%)	A = False B = False
Alt D  9 (64.29%)	None of the above

Questions corrected  
by peers...

### Comments

★★★

Check this page. && is higher than ||.  
<http://java.sun.com/docs/books/tutorial/java/nutsandbolts/operators.html>

so the equation is  $A + B!B + !A$ , which becomes  $A + !A$ , which always evaluates to true.

So it doesnt matter what values you put into A and B, the expression is never going to be false.

★

I think that the && operation has a higher priority and so will be evaluated before the ||  
i.e. `(!A) || (B && (!B)) || (A)`  
havent double-checked in textpad though...

★

As explained by the person above me who linked to the sun page, as that expression stands, it cannot be false. In bracket form it would look like: `(!A || B) && (A || !B)`.

Of the answers you gave, none of the above is the correct one. :P

#### Author's reply

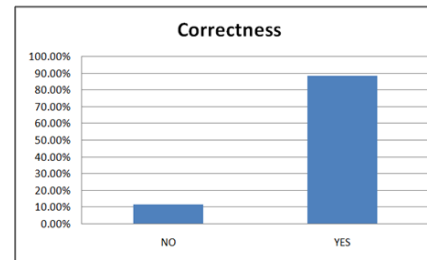
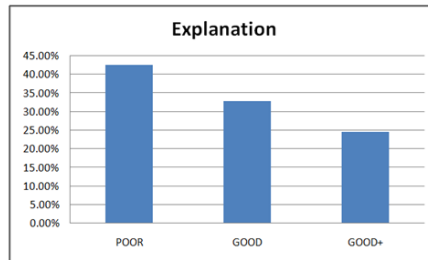
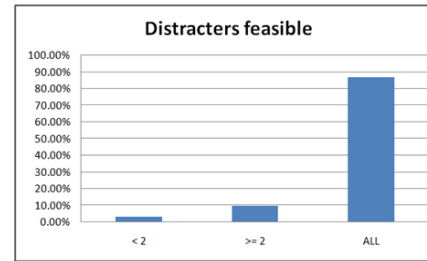
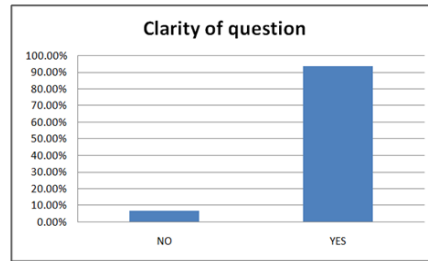
Sorry everyone..thanx for the reply..i've posted the new version of this question. Feel free to check it out n comment on it (i've 'repaired' my understanding, i hope i got it right this time :D)

ya && is at a higher lvl than || so always do && so in this case the answer can only be true no matter what

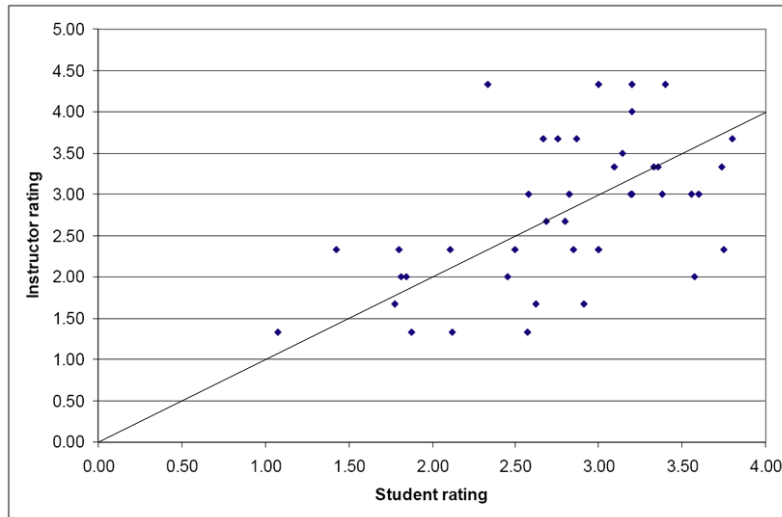
Wow that actually helped me alot lol. Totally forgot about the order of && and ||

Classification  
of 10% of  
repository of  
617  
questions:

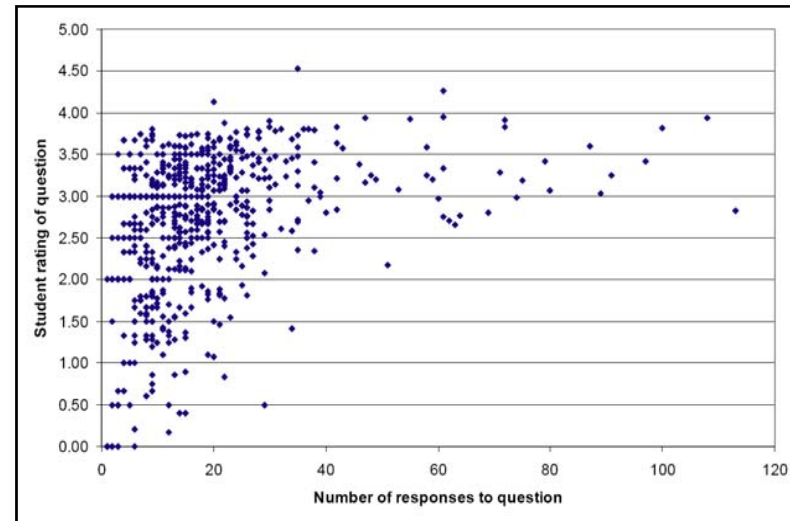
Computer Science (CS1)



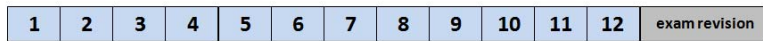
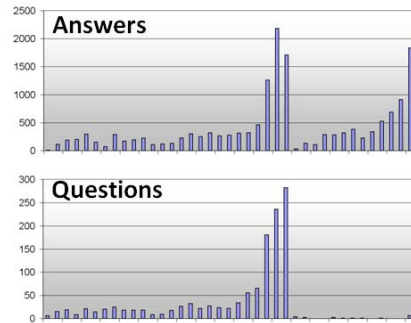
How effectively do students  
rate questions?



Can students avoid poor  
quality questions?



# Efficacy



↑ Mid-term test  
↑ PeerWise introduced  
↑ Assessment deadline  
↑ Final exam

The more “active” students (M) performed better than less “active” students (L) of similar ability (grouped by quartiles, ranked by mid-term mark) in formal examinations:

