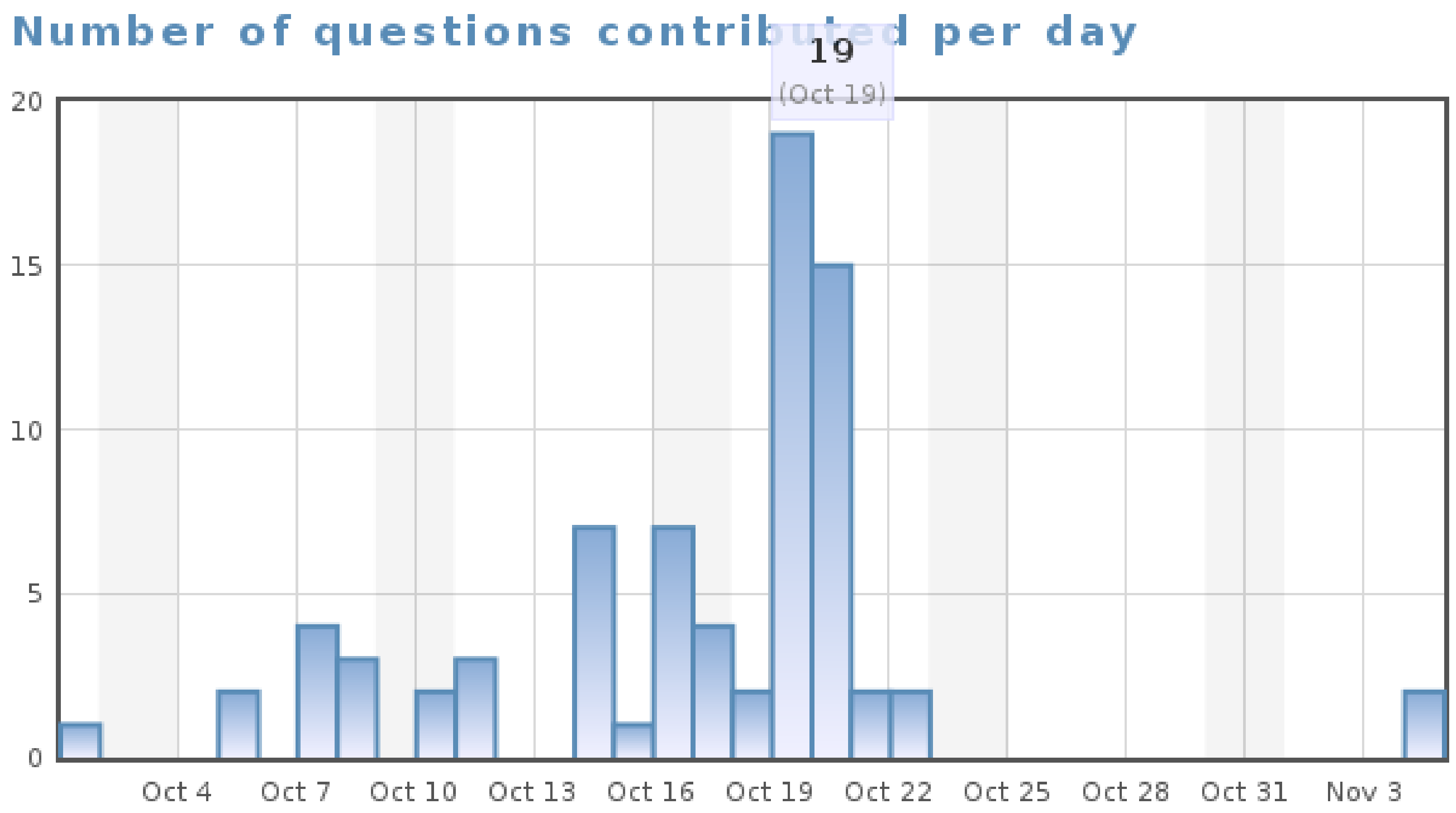
Introduction to Java Programming
Mid-term review

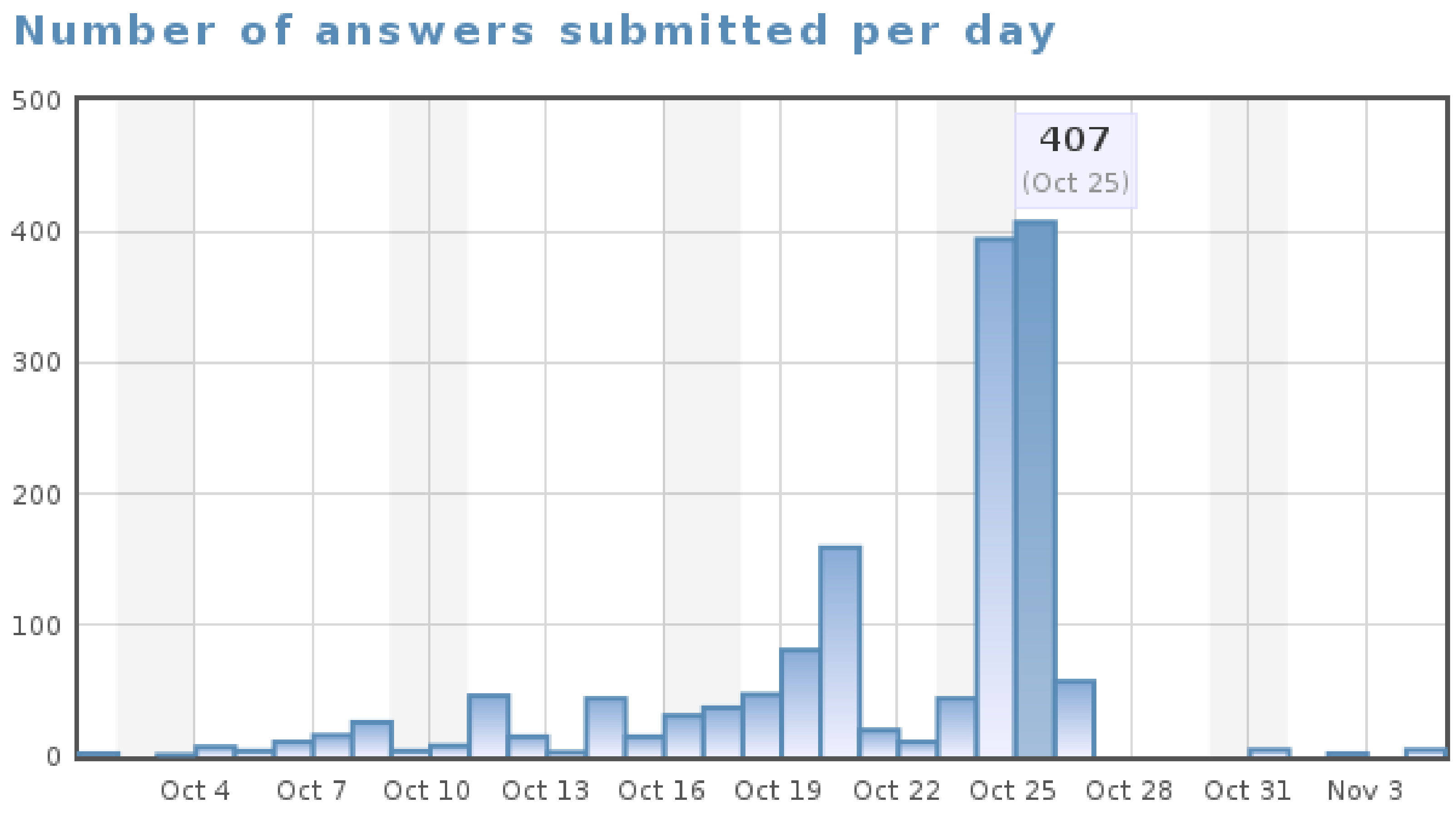
Massimo Felici   Conrad Hughes

http://www.inf.ed.ac.uk/teaching/courses/ijp

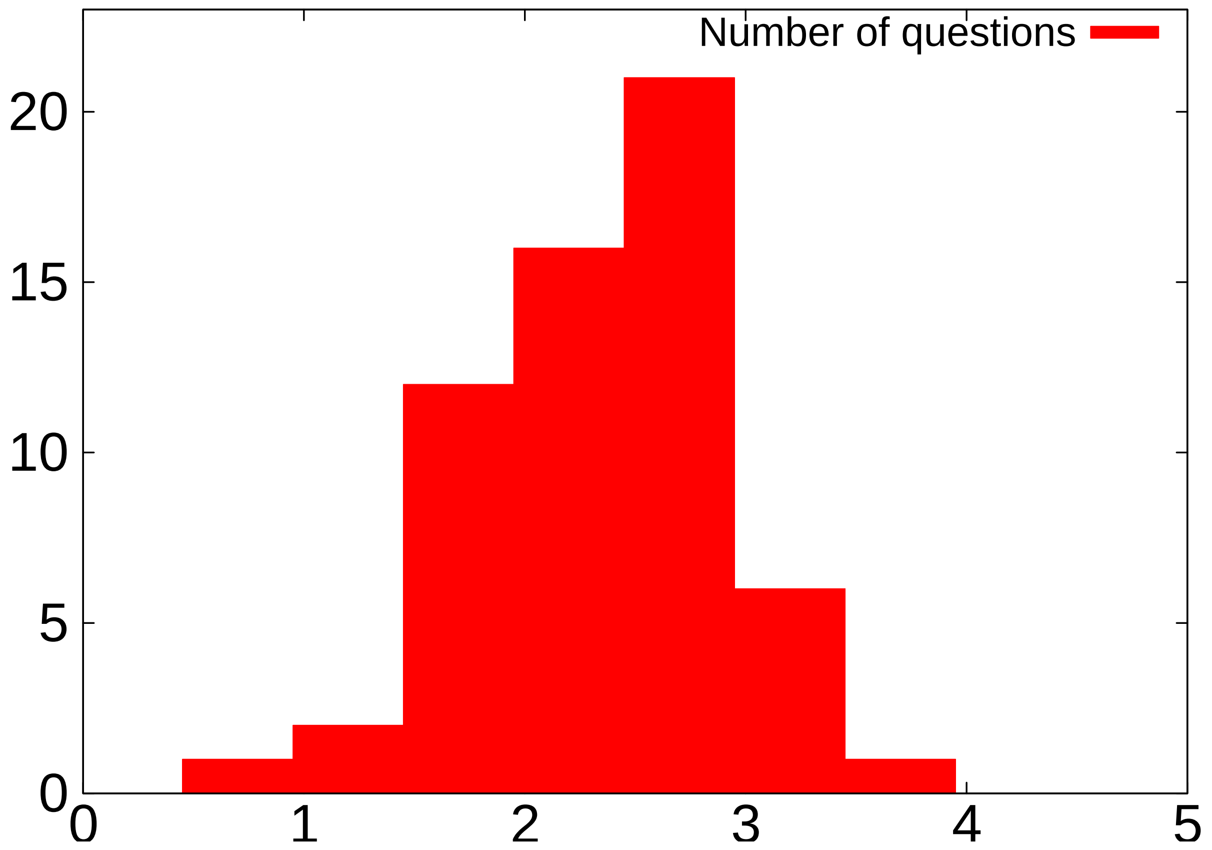
School of informatics

PeerWise


Number of questions contributed per day
19
(Oct 19)
20
15
10
5
0
Oct 4
Oct 7
Oct 10
Oct 13
Oct 16
Oct 19
Oct 22
Oct 25
Oct 28
Oct 31
Nov 3


Number of answers submitted per day
500
400
300
200
100
0
407
(Oct 25)
Oct 4
Oct 7
Oct 10
Oct 13
Oct 16
Oct 19
Oct 22
Oct 25
Oct 28
Oct 31
Nov 3

Question quality



What's bad: questions

- Trick questions (some obscure error that's easy to miss, or appearing to ask about one thing when you're really asking about something different and not obvious).
- Questions that don't make sense (because of errors or ambiguities), or which aren't precise enough to have a straightforward answer.
- Questions that ask something so obscure that you don't learn anything really useful.

What's bad: explanation and answers

- Explanation is missing, too short, or unclear.
- Questions where the “right” answer is wrong (didn't test!).
- Questions where one of the distractors is also right (didn't test distractors).

Badness: best versus the worst

	Best 10 questions	Worst 10 questions
Bad explanation	3	6
Right is wrong	0	6
Wrong is right	1	3
Trick question	1	3
Unclear question	0	3
Obscure question	1	2
Average (over 10)	0.5	2.3

“Best” and “worst” according to your quality ratings.

What’s good

- Focuses on one thing.
- It’s obvious what the subject of the question is.
- All of the distractors are credible (so you have to think!).
- Has a good explanation.
- Makes an important aspect of the language clear.

Goodness: best versus the worst

	Best 10 questions	Worst 10 questions
Focused	8	3
Obvious subject	7	2
Credible answers	7	8
Good explanation	6	1
Enlightening	6	0
Average (over 10)	3.4	1.4

“Best” and “worst” according to your quality ratings.

How do you think Quiz 1 would score?

. . . maybe we should have poll. . .

Try to do the following

- Mainly you should focus on one idea per question: the idea is not to confuse or distract people, but get them to think about some aspect of programming and see if they understand it. If you mix several ideas, then they might get your question wrong for lots of different reasons and feel that you're being unfair.
- Explain clearly why the right answer is correct (and make sure that it is!). Give example code if it's appropriate.
- Explain clearly why the distractors are wrong (and make sure that they are!). Give example code if it's appropriate.

Suggestions

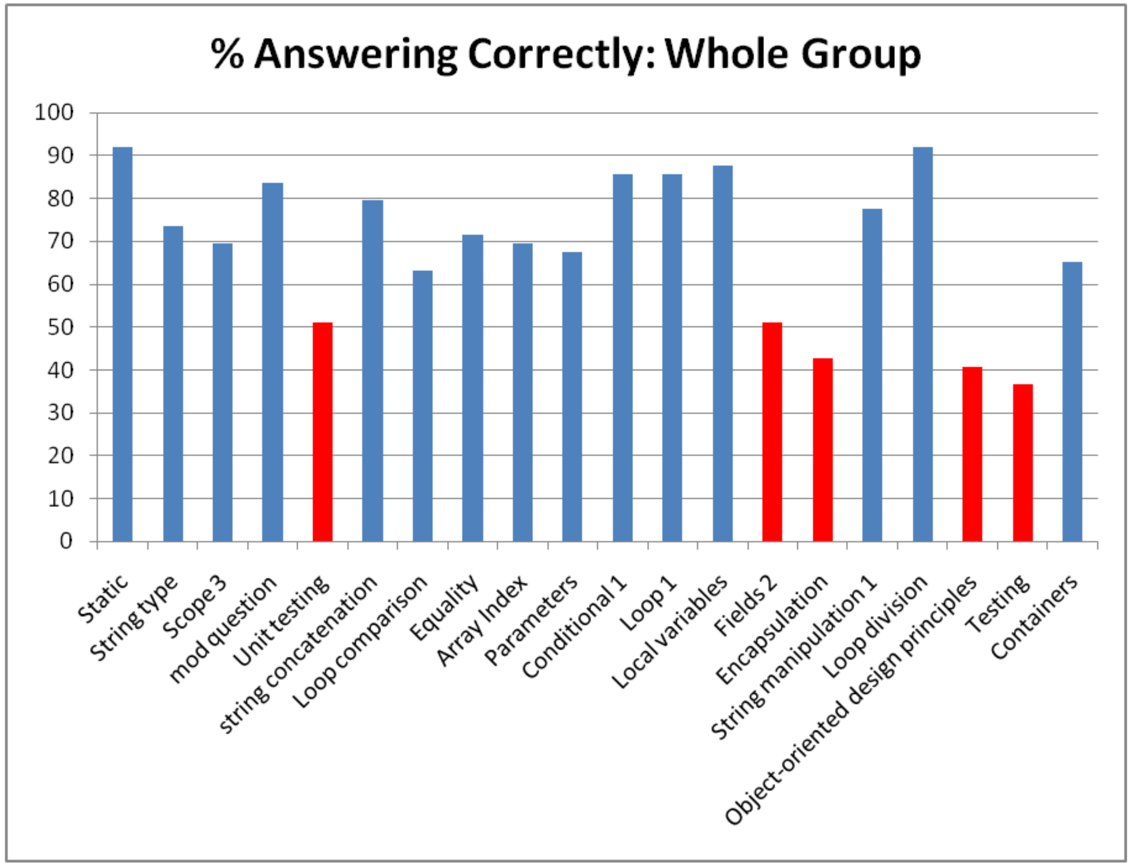
- Sometimes a picture will help. You can also use HTML formatting, such as `<pre>` (for source code).
- Sometimes it will help to include a link to another web page (Wikipedia, for example) which explains the idea in more detail. But do write your own explanation: there's nothing quite like trying to explain an idea for making sure that *you* understand it!
- Run your question past a couple of friends, and see what they think.

Motivation

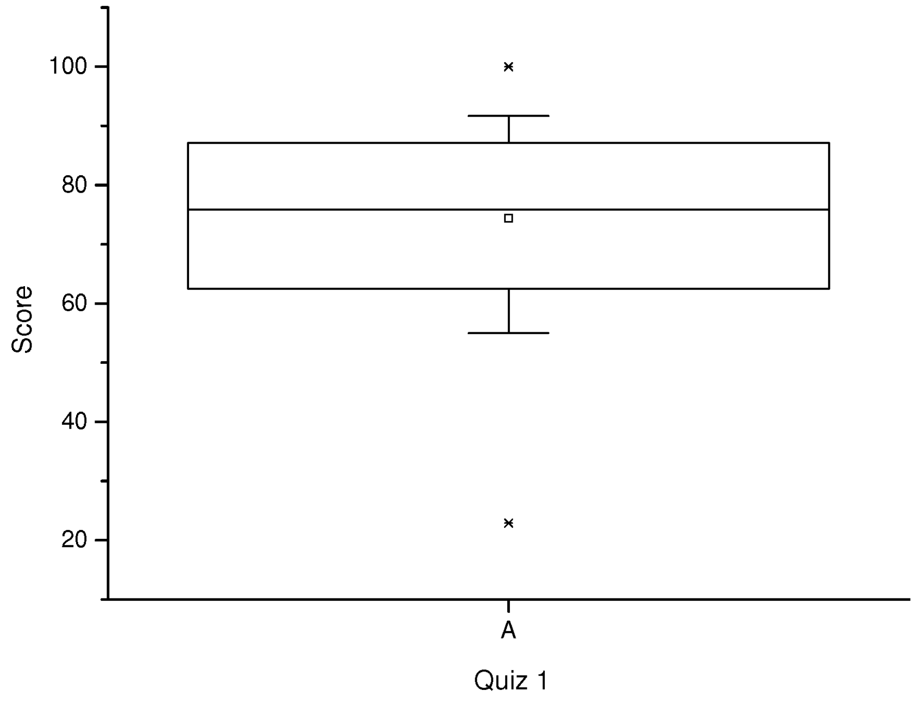
We'll ask for more detailed feedback from you at the end of term, but we are trying out PeerWise as an experiment . . .

- How many of you have found PeerWise somewhat useful so far?
- Do you think it would have been more useful if the questions were better?
- How much more time would it take to make *your* questions better?

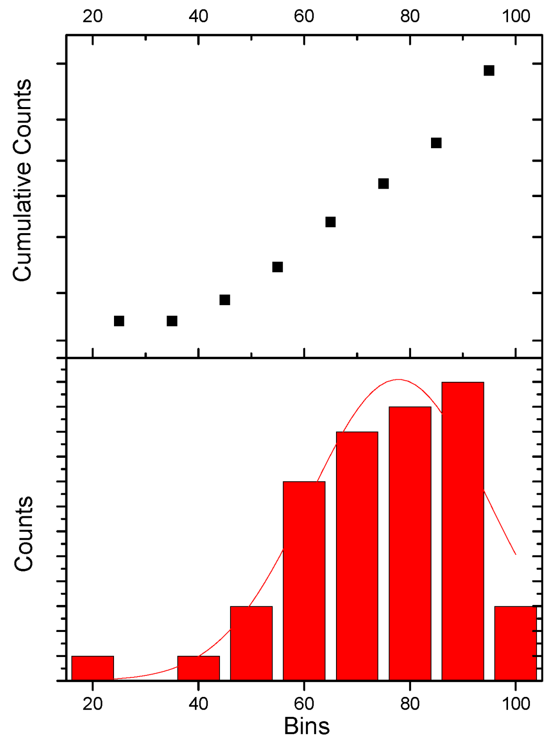
WebCT Quiz 1: Hardest questions



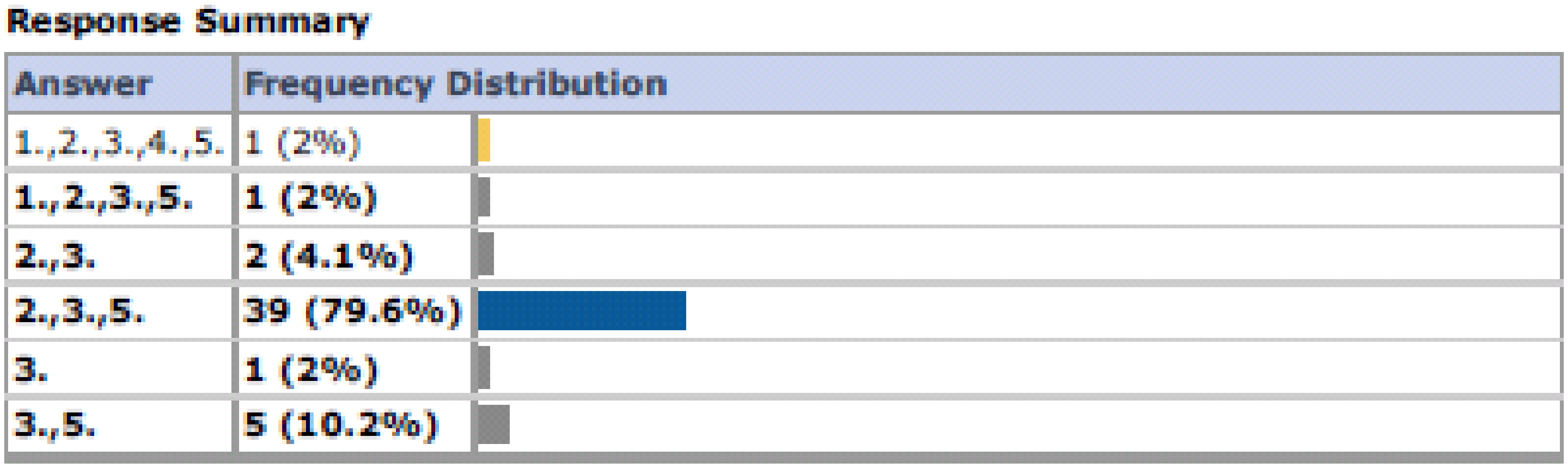
How did you score?



How did you score?



Question: String Concatenation

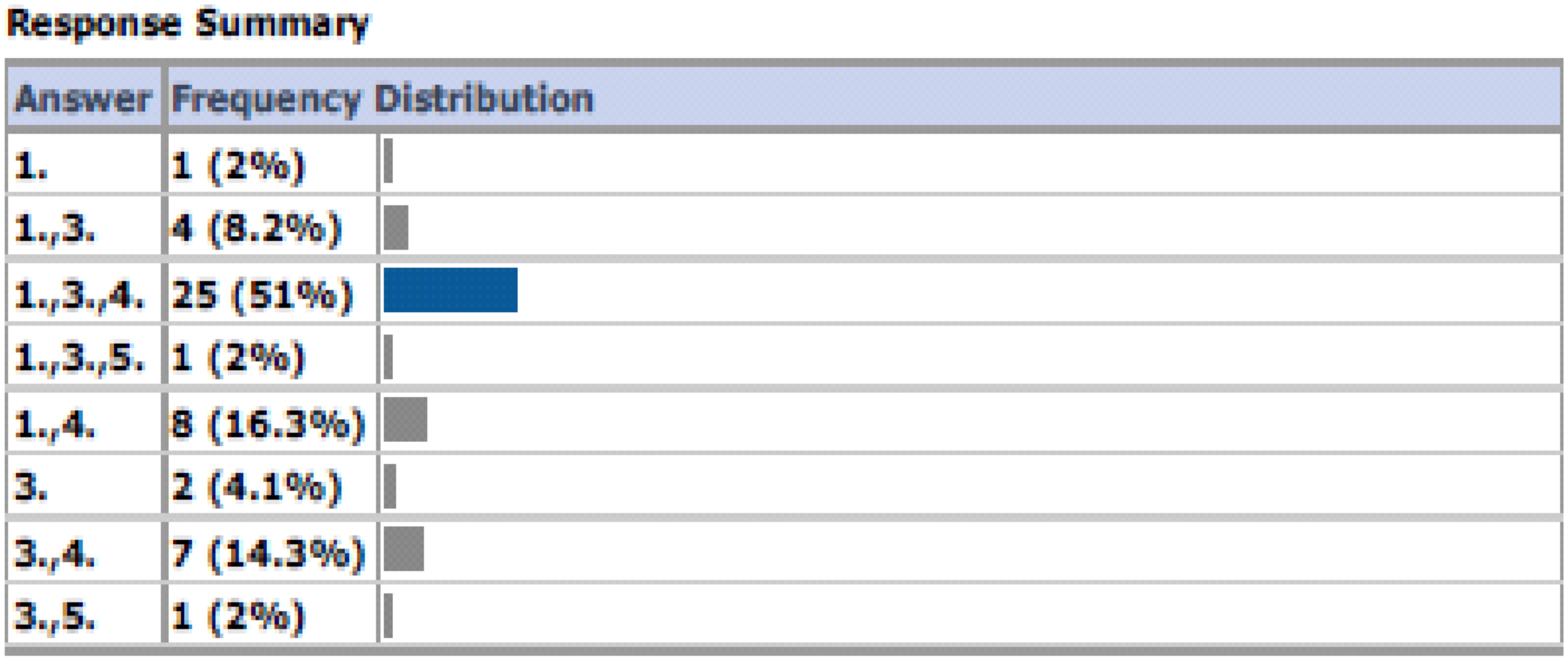


Response Summary

Answer	Frequency Distribution
1.,2.,3.,4.,5.	1 (2%)
1.,2.,3.,5.	1 (2%)
2.,3.	2 (4.1%)
2.,3.,5.	39 (79.6%)
3.	1 (2%)
3.,5.	5 (10.2%)

What's "2" + 2?

Question: Field

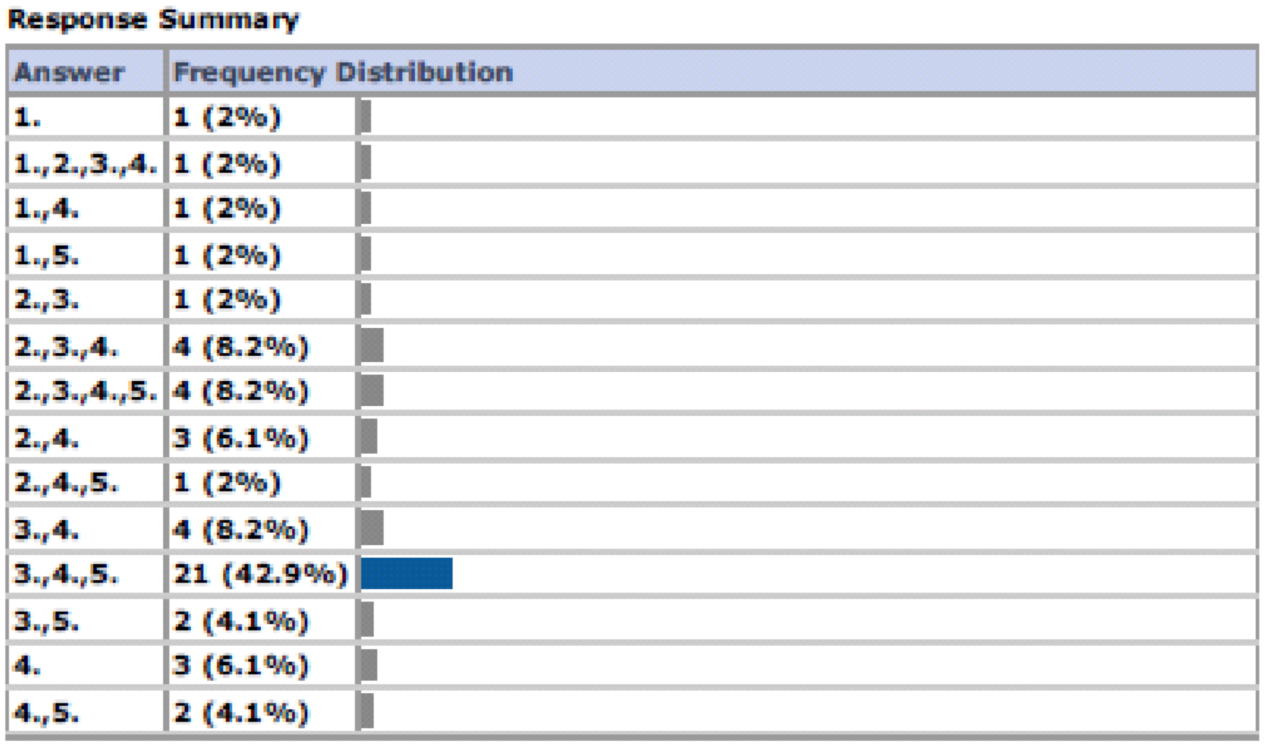


Response Summary

Answer	Frequency Distribution	
1.	1 (2%)	
1.,3.	4 (8.2%)	
1.,3.,4.	25 (51%)	
1.,3.,5.	1 (2%)	
1.,4.	8 (16.3%)	
3.	2 (4.1%)	
3.,4.	7 (14.3%)	
3.,5.	1 (2%)	

Scope, lifetime, purpose.

Question: Encapsulation

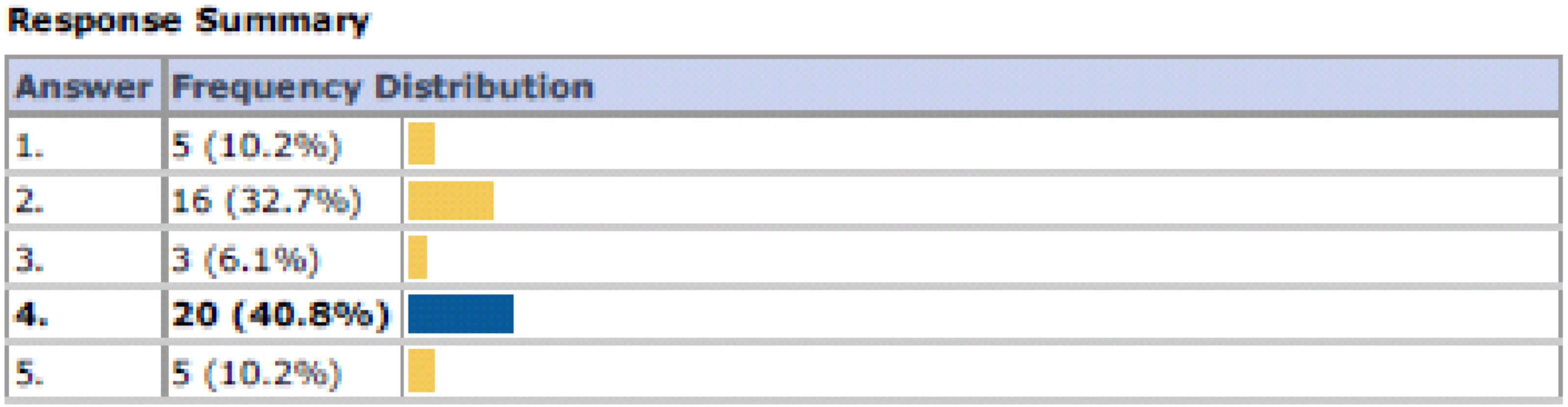


Response Summary

Answer	Frequency Distribution	
1.	1 (2%)	▮
1.,2.,3.,4.	1 (2%)	▮
1.,4.	1 (2%)	▮
1.,5.	1 (2%)	▮
2.,3.	1 (2%)	▮
2.,3.,4.	4 (8.2%)	▮
2.,3.,4.,5.	4 (8.2%)	▮
2.,4.	3 (6.1%)	▮
2.,4.,5.	1 (2%)	▮
3.,4.	4 (8.2%)	▮
3.,4.,5.	21 (42.9%)	▮
3.,5.	2 (4.1%)	▮
4.	3 (6.1%)	▮
4.,5.	2 (4.1%)	▮

Public/private, accessors & mutators, hiding implementation detail.

Question: Object-Oriented Design Principles

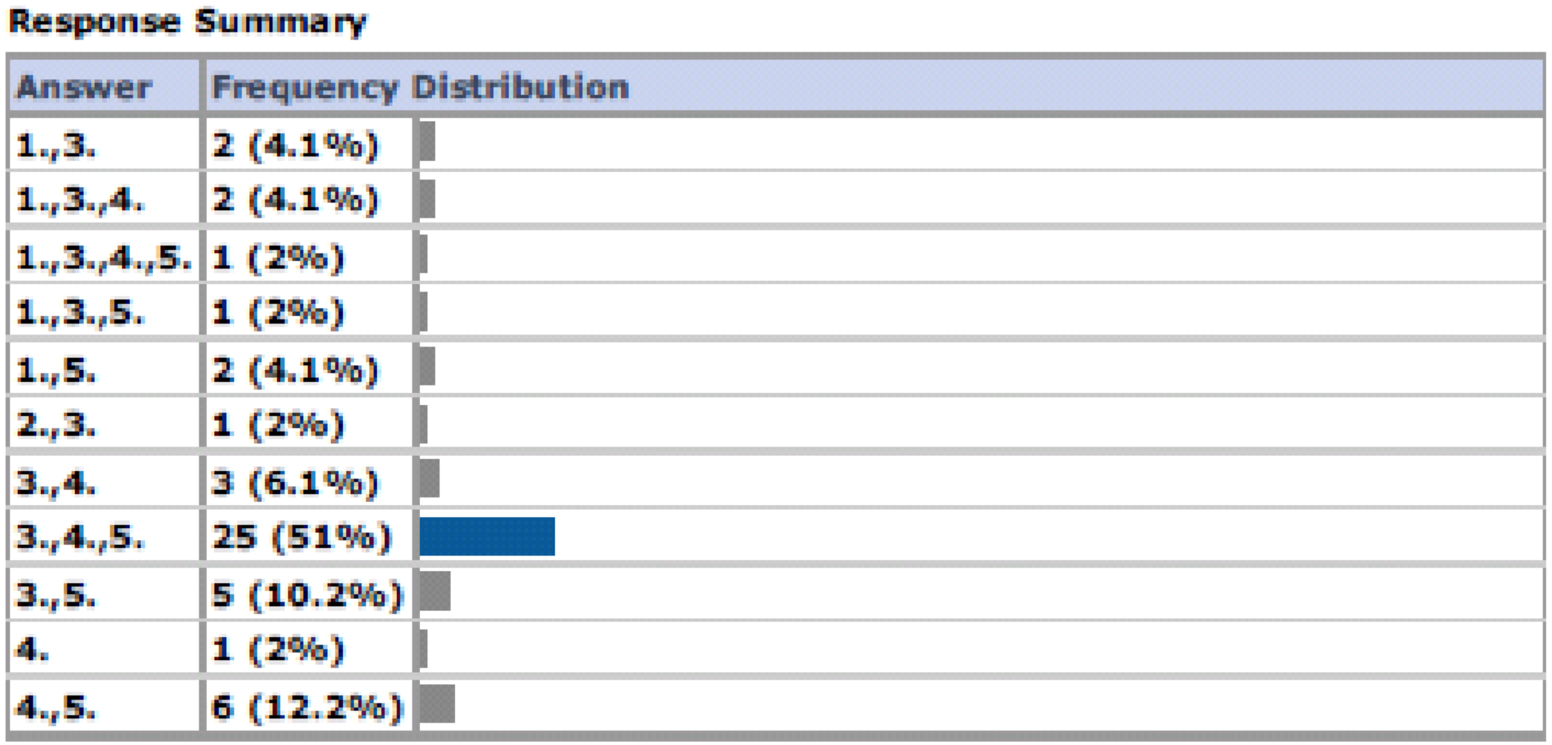


Response Summary

Answer	Frequency Distribution	
1.	5 (10.2%)	▮
2.	16 (32.7%)	▮▮▮
3.	3 (6.1%)	▮
4.	**20 (40.8%)**	▮▮▮▮
5.	5 (10.2%)	▮

Cohesion, coupling, encapsulation.

Question: Unit Testing

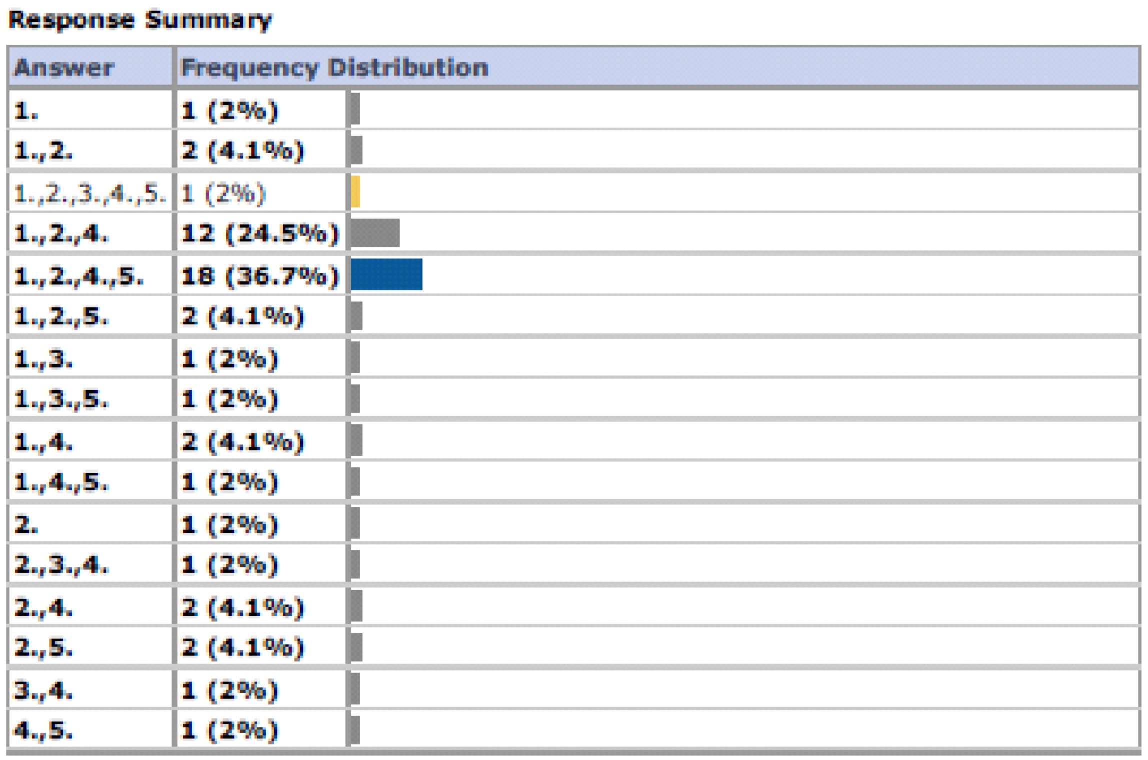


Response Summary

Answer	Frequency Distribution
1.,3.	2 (4.1%)
1.,3.,4.	2 (4.1%)
1.,3.,4.,5.	1 (2%)
1.,3.,5.	1 (2%)
1.,5.	2 (4.1%)
2.,3.	1 (2%)
3.,4.	3 (6.1%)
3.,4.,5.	25 (51%)
3.,5.	5 (10.2%)
4.	1 (2%)
4.,5.	6 (12.2%)

What is a fixture?

Question: Testing



Response Summary

Answer	Frequency Distribution
1.	1 (2%)
1.,2.	2 (4.1%)
1.,2.,3.,4.,5.	1 (2%)
1.,2.,4.	12 (24.5%)
1.,2.,4.,5.	18 (36.7%)
1.,2.,5.	2 (4.1%)
1.,3.	1 (2%)
1.,3.,5.	1 (2%)
1.,4.	2 (4.1%)
1.,4.,5.	1 (2%)
2.	1 (2%)
2.,3.,4.	1 (2%)
2.,4.	2 (4.1%)
2.,5.	2 (4.1%)
3.,4.	1 (2%)
4.,5.	1 (2%)

Positive/negative testing, assertions, walkthroughs, debugging.

“Tick all that apply” vs “Tick only one”

- Tick all that apply (tick 1–5 = full marks!): 82%
- Tick all that apply, with negative marking: 73%
- Tick only one: 77%
- Tick only one, with negative marking *(estimate)*: 69%

. . . “Tick all that apply” *looks* harder, but actually on average it isn’t, because you get partial credit.

Concluding remarks on Quiz 1

- Main issues were with **Object-Oriented** concepts and **Testing**, the two chapters just before the test.
- Fairly easy (that is, marks of 70% and above) for the majority.

Assignment 1

We're still marking it, sorry! :(You'll all get an individual feedback email next week. Here are some issues we've seen so far:

- Not focusing all the work related to a particular task in one place (*coherence*), or using the right type of variable (local vs instance vs class/static).
- Tests that aren't tests (always `assert`). Sometimes it's difficult to come up with the right assertion, but you'll get better at that with practice!

This course

This course is not just about getting marks! It may appear to be an easy pass (average 64%–67% over the last four years, with a very low standard deviation: 5%–12%), but that's not all that it's about: programming is almost certainly an essential skill that will make it far easier for you to do well in your other course work and your MSc. projects. Since we don't have an exam after term, your notional 100 hours should go entirely into study and project work, all of which will be over on the 3rd of December, before any of your exams!

The book

Do read the book and try as many exercises as you can! Programming skills really improve with practice, and the book does (we think) do a pretty good job of introducing lots of important OO ideas.

Assignment 2

The earliest adventure games, as documented on the Wikipedia page, were text-only, such as your starting point, Zuul. To give you some ideas regarding interface, here are a couple of examples from history.

The Hobbit (1982): This was one of the first adventure games to mix text and graphics. First game I ever played too!

`http://www.lysator.liu.se/tolkien-games/entry/hobbit.html`

`http://www.youtube.com/watch?v=SZsyv5aKw4U`

Myst (1993): Another landmark in gaming history, Myst's interface is purely graphical, based on a (static) "first person" view: you move by clicking on exits, and interact with objects in the current room by clicking and dragging them.

`http://www.youtube.com/watch?v=cIRtutbSwak`
`http://www.youtube.com/watch?v=zJNlAl9cxdY`

. . . In between those two there were games with menus and buttons, but I can't remember their names I'm afraid. Obviously things have changed a lot in the two decades since. These examples are **well beyond what we're expecting** (Myst cost around $500,000 to make I think), but they might give you some ideas for things to try.

Assignment 2

. . . any questions..?