Foundations of CS

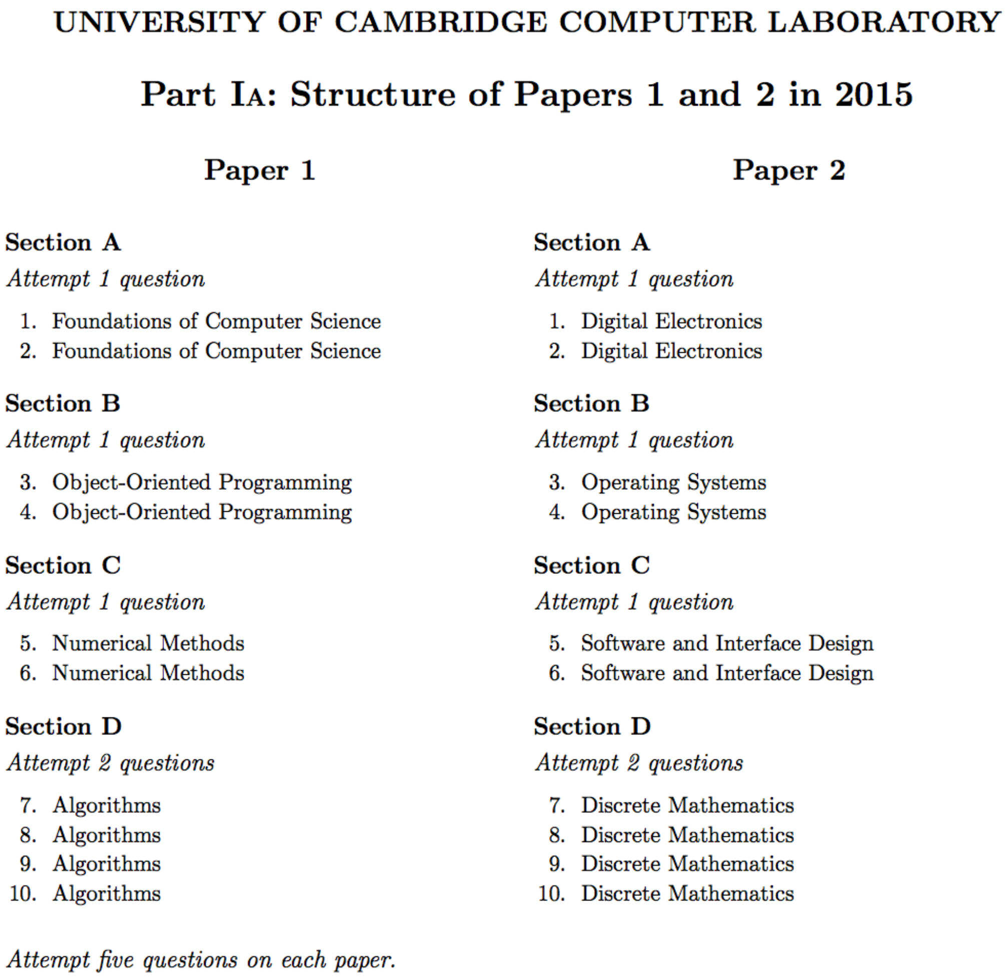
Example Class 1

these slides are online at http://hh360.user.srcf.net/blog/teaching

Aim of today

- inform you about the very basics of the 1A tripos exam
- give you my opinion (!!) on how to approach ML problems
- highlight the most common mistakes I see in students work
- get you into good habits early
- teach you that ML is best language in the world :-)

All of this is my personal opinion, if anyone else contradicts me, trust them not me



UNIVERSITY OF CAMBRIDGE COMPUTER LABORATORY

Part IA: Structure of Papers 1 and 2 in 2015

Paper 1

Section A

Attempt 1 question

1. Foundations of Computer Science
2. Foundations of Computer Science

Section B

Attempt 1 question

3. Object-Oriented Programming
4. Object-Oriented Programming

Section C

Attempt 1 question

5. Numerical Methods
6. Numerical Methods

Section D

Attempt 2 questions

7. Algorithms
8. Algorithms
9. Algorithms
10. Algorithms

Paper 2

Section A

Attempt 1 question

1. Digital Electronics
2. Digital Electronics

Section B

Attempt 1 question

3. Operating Systems
4. Operating Systems

Section C

Attempt 1 question

5. Software and Interface Design
6. Software and Interface Design

Section D

Attempt 2 questions

7. Discrete Mathematics
8. Discrete Mathematics
9. Discrete Mathematics
10. Discrete Mathematics

Attempt five questions on each paper.

taken from http://www.cl.cam.ac.uk/teaching/exams/exam-structure.pdf

Timing

- Each paper is 3 hrs long and has 5 question
- After 30 mins for picking questions and checking, that leaves 30 mins per question
- Each question is 20 marks so 1.5 mins per mark

Approach

Focus on the main function, assume auxiliary functions and write them later

DO NOTs

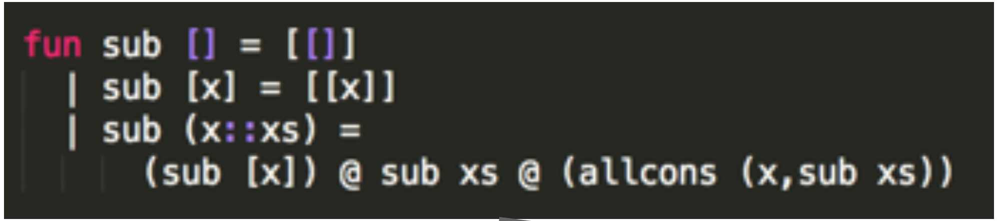
To avoid:

Please try to avoid the following:

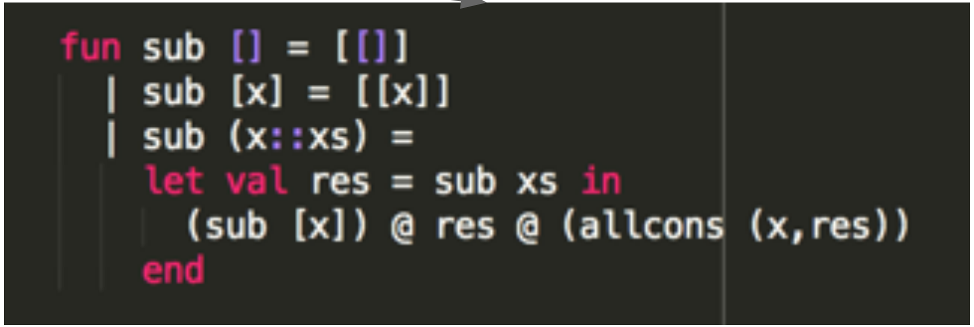
- binding to variables that you do not use (wildcard is your friend here)
- pattern match cases that will never be matched
- if x then true else false or if x=true then ... else ...
- strange auxiliary functions when a standard function will do
- using cons (e.g. x::xs) and then only using the list as whole (e.g. always referring to x::xs never just x or xs)

To avoid (2):

- recomputing results



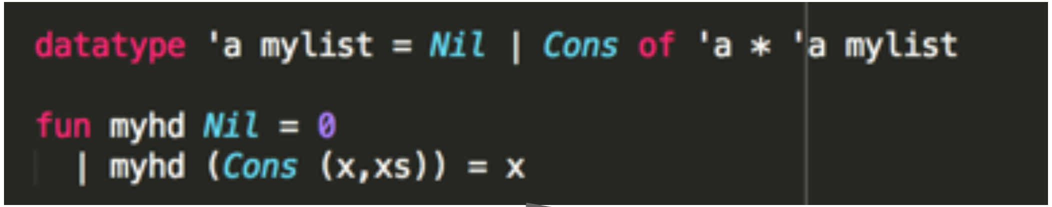
```
fun sub [] = [[]]
  | sub [x] = [[x]]
  | sub (x::xs) =
    (sub [x]) @ sub xs @ (allcons (x,sub xs))
```



```
fun sub [] = [[]]
  | sub [x] = [[x]]
  | sub (x::xs) =
  let val res = sub xs in
    (sub [x]) @ res @ (allcons (x,res))
  end
```

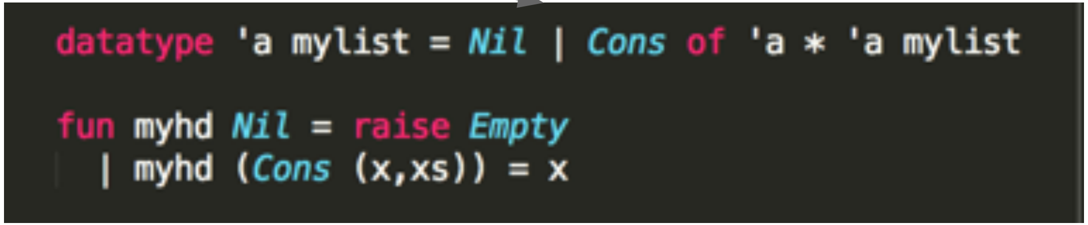
To avoid (3):

- unnecessary constraints on the type of a function that should be polymorphic



```
datatype 'a mylist = Nil | Cons of 'a * 'a mylist

fun myhd Nil = 0
  | myhd (Cons (x,xs)) = x
```



```
datatype 'a mylist = Nil | Cons of 'a * 'a mylist

fun myhd Nil = raise Empty
  | myhd (Cons (x,xs)) = x
```

To avoid (4):

- using if-then-else when pattern matching could be used
- nesting functions unnecessarily (if they don’t use locally scoped variables then they don’t need to be nested
- trying to hide issues with your code from the examiner (or me), acknowledge and state how you might address them

DOs

To do:

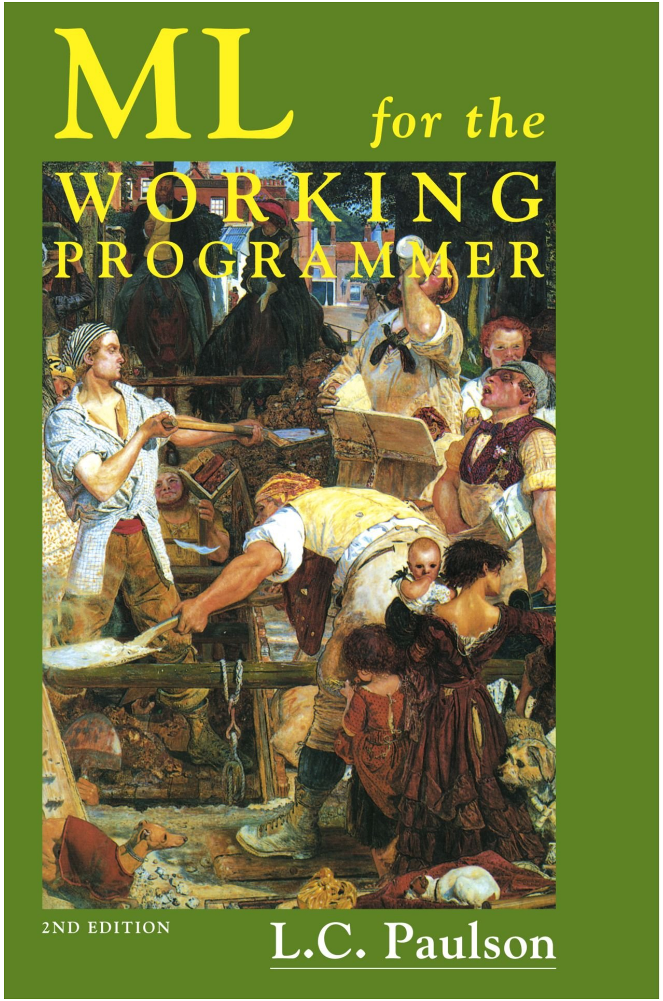
Please do:

- Read the whole questions to start with
- Reuse functions from previous parts of the question
- Use diagrams, particularly for binary tree question
- Include call traces for sample input
- State the type of the your functions (don’t forget about equality type)
- Wrap your function to initialise values such as accumulators

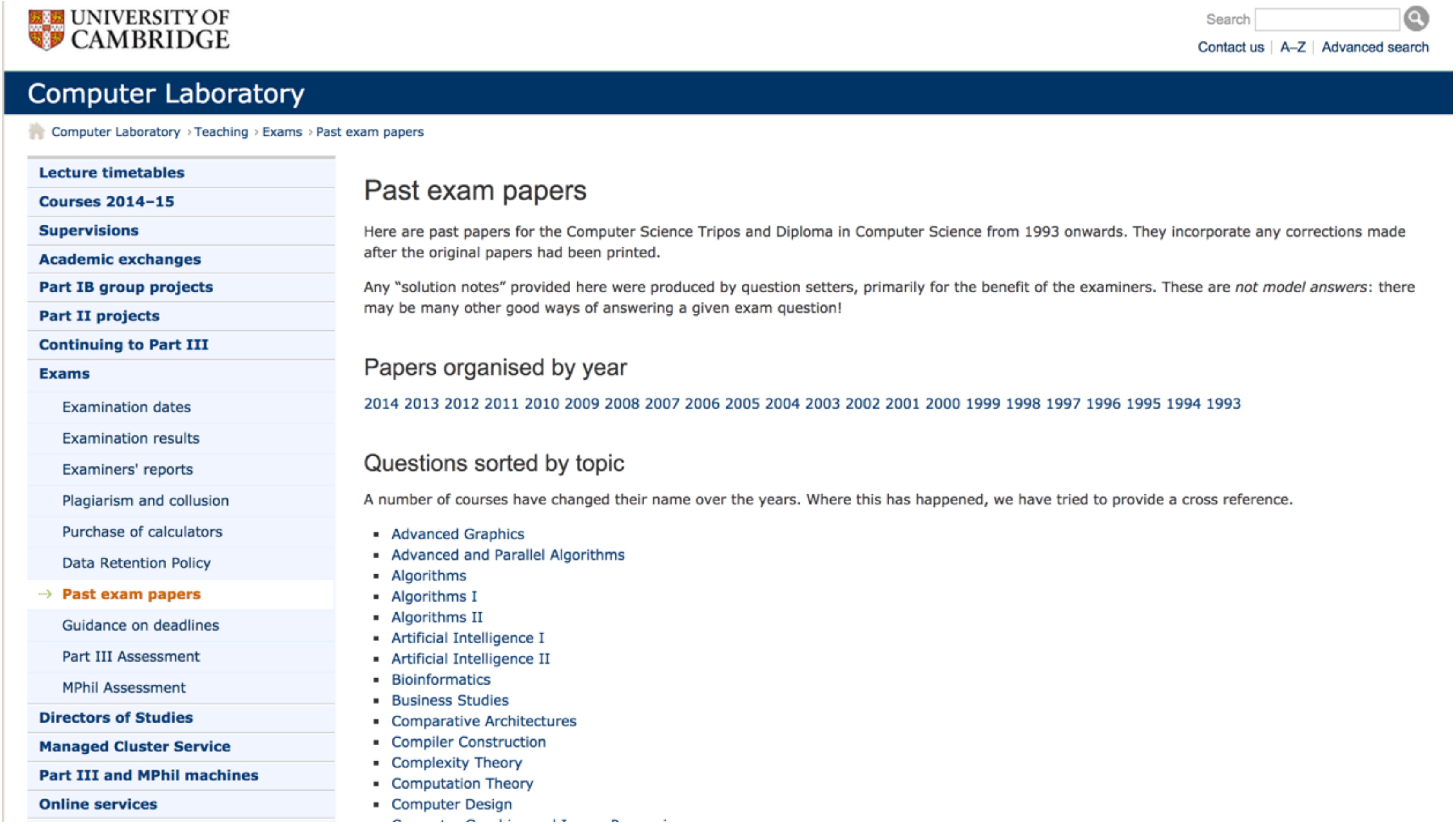
To do (2):

- Use big-O notation when talking about efficiency
- Comment on both space and time efficiency
- Justify your answers (using recurrence relations if you can)
- Write functions with multiple arguments (now that you know how)
- Throw useful exceptions: give them sensible names and add values to them

RESOURCES



An excellent source of interesting problems and follows the course quite closely



UNIVERSITY OF CAMBRIDGE

Search | Contact us | A–Z | Advanced search

Computer Laboratory

Computer Laboratory › Teaching › Exams › Past exam papers

- Lecture timetables
- Courses 2014–15
- Supervisions
- Academic exchanges
- Part IB group projects
- Part II projects
- Continuing to Part III
- Exams
  - Examination dates
  - Examination results
  - Examiners' reports
  - Plagiarism and collusion
  - Purchase of calculators
  - Data Retention Policy
  - Past exam papers
  - Guidance on deadlines
  - Part III Assessment
  - MPhil Assessment
- Directors of Studies
- Managed Cluster Service
- Part III and MPhil machines
- Online services

Past exam papers

Here are past papers for the Computer Science Tripos and Diploma in Computer Science from 1993 onwards. They incorporate any corrections made after the original papers had been printed.

Any "solution notes" provided here were produced by question setters, primarily for the benefit of the examiners. These are *not model answers*: there may be many other good ways of answering a given exam question!

Papers organised by year

2014 2013 2012 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993

Questions sorted by topic

A number of courses have changed their name over the years. Where this has happened, we have tried to provide a cross reference.

- Advanced Graphics
- Advanced and Parallel Algorithms
- Algorithms
- Algorithms I
- Algorithms II
- Artificial Intelligence I
- Artificial Intelligence II
- Bioinformatics
- Business Studies
- Comparative Architectures
- Compiler Construction
- Complexity Theory
- Computation Theory
- Computer Design

TRIPOS QUESTION TIME

Foundations of Computer Science 2005 – Paper 1 Question 5 (LCP)

(*a*) Explain the operation of the Quicksort algorithm. Illustrate your answer by applying it to the list $[8, 3, 6, 12, 2, 9, 20, 1, 5, 0, 7, 13, 4, 11, 10]$. [6 marks]

(*b*) Write a Standard ML function for finding the median of three integers. [3 marks]

(*c*) A variant of Quicksort uses a novel method of choosing the pivot element. Instead of using the head of the list, it uses the median of the first, middle and last elements of the list. Express this algorithm in ML. You may assume the existence of the function `length`, but you may not assume that the items being sorted are distinct. [7 marks]

(*d*) What is the average-case execution cost, measured in terms of the number of comparisons, for the version of Quicksort described in (c) above? Justify your answer carefully. [4 marks]