- CS4112 Computer Science 2
- Lecturer: Michael English
- email: Michael.English@ul.ie
- Office: CS1-024
- Tel: 061 202772
- Lecture hours: Tuesday 1pm SG16; Wednesday 1pm SG16
- Tutorials start Week 2

- Course Material available on www.csis.ul.ie/
  - Follow link to Current Students etc...
- Alternatively go directly to: http://www.csis.ul.ie/coursemodule/CS4112

- Recommended Texts: (Available in the library)

  1. Logic and Discrete Mathematics: A Computer Science Perspective
     Authors: W.K. Grassman and J.P. Tremblay

  2. Discrete Mathematics and Its Applications by Kenneth Rosen

  3. Introductory Java Programming Books in Library

  4. Section 510/511 in Library: Discrete Mathematics

  5. Lecture Notes from Imperative Programming 1

- Course Assessment

  – 2 Mid-Term exams: Week 5(15%), Week 9(15%)

  – End-of-Semester exam 70%

- GRADING

  – A1>=80, A2>=72

  – B1>=64, B2>=60, B3>=56

  – C1>=52, C2>=48, C3>=40

  – D1>=35, D2>=30

- Course Outline

- CS4111 Computer Science 1

- How to represent a problem in such a way that a computer can solve it

- Writing programs that implement solutions to these problems

- Expressions → mathematical functions → lambda calculus → Scheme

- Design of programming languages: any program that can be solved by a computer must have a solution which can be programmed in the programming language

- Natural languages: too imprecise, ambiguous

- Mathematical notation: problems can be formulated which cannot be solved by computer

- Programming Paradigms: style of programming; a way of thinking about programming

- Imperative Programming: characterised by commands that update(access) variables

- Functional Programming: based on functions and expressions

- Object-Oriented Programming: based on the idea of grouping data and the functions that operate on that data into units called objects which are described by classes

- Computer Science 2

- Equivalence of mathematical function and a computer program

- INPUT → manipulate by computer program → OUTPUT

- INPUT → manipulate by mathematical function → OUTPUT

- A tiny detail can crash a program or cause it to misbehave
- Aircraft, Automotive or Medical Software etc..
- Cost + time + prestige
- Code Inspections
- Program testing: 300 runs failed to uncover the problem
- "Program testing can be used to show the presence of bugs but not their absence"
- PREVENTION is better than CURE

- Why use Mathematics in the design of software?

  1. Allow precise specification of requirements
  2. Translate requirements into programs
  3. Reason about properties of programs

- Therefore mathematics allows us to show that a program performs as expected

- Other topics in course:
  - Under the bonnet of programming languages
  - Choosing the appropriate data structure to solve a problem: performance and efficeincy issues.
  - Introduction to grammars
  - Notations for describing grammars

- A grammar for a natural language
- $< sentence >:< personalPronoun >< verb >< noun >$
- $< personalPronoun >: john|mary$
- $< verb >: walked|went$
- $< noun >: home$
- Similarily for programming language
- $< assignment >:< variable ><=>< value >$
- $< variable >: sum|x|salary$
- $< value >:< int\_part >< . >< real\_part >$