Computing Concepts and Paradigms

Lecturer: Joseph Cordina
CSA1020
Department of Computer Science and A.I.
Rm 203, New Computing Building
joseph.cordina@um.edu.mt
www.cs.um.edu.mt/~jcor1
Course Introduction

• This course will provide an introduction to Computer Science.
• Course is split up in lectures followed by a test (100%) and no assignment.
• You are encouraged to work and read at home
• These course notes will be provided in some format.
• Course builds up on knowledge of previous lectures so be sure to cover everything by the next lecture.
• Lecture attendance is not compulsory but lecture notes are definitely not complete. I encourage that you take notes in class and listen, not hear.
• Please ask and interact. Its for your own good.
Textbooks

- Several others you might feel like reading.
End Slide

• Exam:
  – 3 questions choose 2
  – Each question will be concerned with a specific area we will cover
  – Each question will be made up of several sub-questions
  – You get >45% and you pass
Welcome

- Welcome to what might be 4 exciting years learning new things.
- Main aim of these 4 years: get out and tackle anything that you might encounter in the big bad world.
- You can finish your course in 2 ways:
  - Pass Exams
  - **Understand** and pass your exams
- Forget what you know, this is a process of unlearning.
What is Computer Science?

• The science of programming, more specifically the science of algorithms
• Is it a science like:
  – Physics
  – Chemistry, etc?

• The scientific method
  – Observe some aspect of the universe.
  – Invent a tentative description, called a hypothesis, that is consistent with what you have observed.
  – Use the hypothesis to make predictions.
  – Test those predictions by experiments or further observations and modify the hypothesis in the light of your results.
  – Repeat above 2 steps until there are no discrepancies between theory and experiment and/or observation.

• Example: Cakes!!
Algorithms

• In computer science, an algorithm is a recipe to solve some problem.
• The area of application is very varied.
• An algorithm is a well-defined finite number of steps that takes an input and produces an output.
• What are the inputs and outputs of
  – A recipe?
  – Changing a flat tyre?
  – Looking up a telephone number?
• The inputs and outputs need to be legal and well formalised but not necessarily bounded.
• The algorithmic problem is the mapping of inputs to outputs, the solution is the algorithm.
History

• First algorithm was gcd by Euclid (circa 300 B.C.)
• 1801 first machine (weaving loom) to execute an algorithm (J. Jacquard).
• Babbage (1833) conceived the analytical machine but had no electricity
• Key mathematicians formulated fundamental concepts: Turing, Godel, Markov, Church, Post and Kleene.
• Computer Science is the continuation of these theories.
Questions and Answers

• Building algorithms is one of the hardest, if not the hardest, mental process.

• Questions:
  – Is an algorithm correct to all possible inputs and outputs?
  – Does an algorithm terminate?
  – How long does an algorithm take to terminate?
    • Is it relevant with Moore’s Law?
  – Is an algorithm possible at all?

• Teaser: Does a bag with all numbers have the same number of elements as a bag with even numbers?
Levels of Detail

- To compute an algorithm, the computing machine must understand the language of the algorithm.
- Computers are fairly stupid, they know only a very basic language.
- The purpose of programming languages is to define a base level in which algorithms can be written.
- A programming language needs to be universal enough to allow all types of algorithms to be written in it.
- A programming language needs to be as unambiguous as possible to allow proper execution of the algorithm.
- All programming languages are universal: termed as Turing powerful.
- Why do we have many programming languages?
Algorithmic Constructs

• What are the base necessities to build an algorithm?
  – Direct Sequencing
  – Conditional Branching

• To deal with varying number of inputs we need looping constructs:
  – Bounded Iteration
  – Unbounded Iteration (Conditional interation)

• By combining these control structures we can build any algorithm.

• Another is sometimes seen : goto (yuck)
Algorithm Representations

• Algorithms can be represented in various ways:
  – Pseudo code
  – Flow Charts
  – State Transition Diagrams
  – Data Flow Diagrams, etc.