Lecture 1:

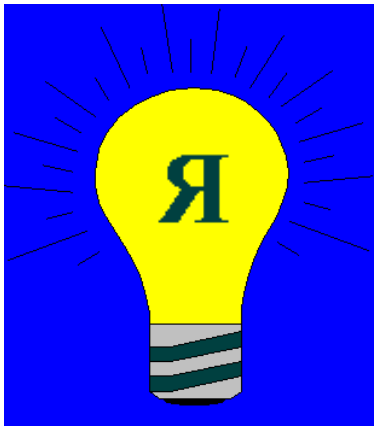
Engineering, Problem Solving, Computers, and Programming

Prof. Shervin Shirmohammadi
University of Ottawa

What is Engineering?

- Legal:
  - Self-Regulating profession: P.Eng.
  - Similar status as Law and Medicine

- Practical:
  - Problem solving

Solutions  **Us**

Aspects Covered

- Technology

  Can we do it?

- Economics

  Can we do it at the right price?

- Society

  Can we do it without harm?

- Communications

  How do we explain ourselves?

Engineering Disciplines

Chemical Engineering

- Techniques mainly from Chemistry, also uses physics, biology, and mathematics.
- Continuous processes to make product-grade quantities of Gasoline, plastics, fertilizers, foods drugs (Biotechnology), nanotechnology, fuel cells, etc.
  - Converting raw material into more useful forms and products.
- Care taken to ensure products and manufacturing are safe.

Civil Engineering

- Provide society with infrastructure:
  - Buildings
  - Bridges
  - Cities
  - Highways
  - Dams
- Uses chemical processes, for example to provide clean water.
- Uses physical processes, for example to erect buildings and bridges.

Computer Engineering

- Application of engineering principles to build computers or computing systems and devices
- individual systems to large ones, from iPad to supercomputers
- Co-design of hardware/software and performance optimization.
- Also deals with things like computer networks (Internet), computer communications, etc.

Electrical Engineering

- Uses electricity, electronics and electromagnetism to build various systems and devices.
- Also uses physical and chemical processes to generate and use electricity.
- Electrical:
  - transmission, motors and machines
  - Reusable energy sources and green technologies
- Electronic:
  - Audiovisual devices, amplifiers, communication systems, computers, etc.

Mechanical Engineering

- Use mostly physical and material science processes (also some chemistry) to build machines, materials, and structures.
- Automotive, aeronautical, locomotive and marine.
- Engines, energy systems.
- Manufacturing, robotics, control.

Software Engineering

- Newest engineering discipline.
  - Much needed due to huge demand!
  - Often ranked number 1 in terms of best careers in North America.
- Uses engineering principles to build software systems and products
- From very small (software in your Coke vending machine) to very large (distributed Amazon cloud) software systems.
- Software is embedded in all modern systems, even cars.
- Application of engineering principles to ensure public safety and warranties.

Problem Solving

Problem Solving

- Define the Problem.
- Gather Information.
- Generate a Possible Solution.
- Implement the Solution.
- Test the Solution.

Define the Problem

- The proper definition of the problem is essential to a good solution.
- Not always easy to see the real problem.
- Breakthroughs come in seeing the real problem.

Beware of the Wrong Problem

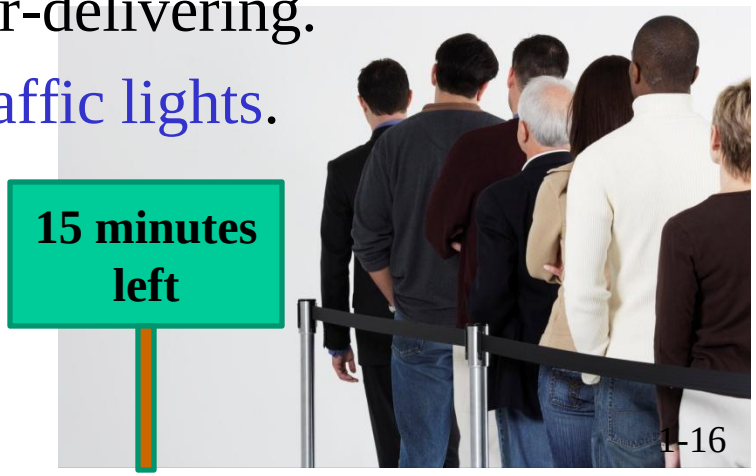
- The bear
- Long lineups
- Slow elevators

The Bear

- Mike and Jack are hiking in the woods when suddenly a bear attacks them, forcing them to climb a tree. As the bear starts to climb the tree too, it becomes obvious that Mike and Jack soon need to jump down and make a run for it. Jack starts putting on his running shoes:
- Mike: You are an idiot, Jack! There is no way those running shoes are gonna help you outrun the bear!
- Jack: Actually, Mike, I don't need to outrun the bear, I just need to outrun you ;)

Long Lineups

- An amusement park receives lots of complaints that their lineups are too long.
- They increase their ride infrastructure and personnel to speed up the queues, but complaints are still coming in.
- One of their engineers recommends the following solution, which works and significantly reduces the number of complaints:
- At specific distances in the line, put a sign saying how many minutes are left to the front of the line. But instead of putting the right value, put a value that is 1.5 times the expected; i.e., manipulate people's expectation by under-promising and over-delivering.
- Similar problem and solution for long traffic lights.



Slow Elevators

- The management of a high-rise building has been receiving too many complaints from occupants that the elevators are slow. They ask the elevator company for help.
- After huge expenses, the elevators are working faster, but people are still complaining. The company says that given the infrastructure of the building and the number of occupants on each floor, the elevators are as fast as they can be.
- The management then asks one of their engineer occupants for help, who after studying the case and interviewing occupants, recommends that mirrors be installed around elevator door frames. The number of complaints significantly decrease after mirrors are installed.
- Why? Because the real problem is people's impatience. Give people something to do (like fix themselves in the mirror) and they are distracted for a short while, making the wait time appear less.
  - **Time flies when you're having fun.**
  - **A watched pot never boils.**

Problem Identification

- What is the problem reported?
  - Why is it a problem?
    - And why is that a problem?
      - …
- There might be many solutions to a problem:
  - Bear: Design and build a shoe that allows you to run faster than a bear!
  - Lineup: Increase the ride capacity and hire more personnel.
  - Elevator: Increase the number of elevators in the building, and also make each of them faster.
  - But these are all very very expensive solutions!
- An engineer will always consider alternatives, not just in his/her design and solution, but starting with the problem itself!

Gather Information

- What are the constraints?
- What are the physical laws?
- What are the mathematical equations?

Generate a Possible Solution

- There may be many possible solutions - pick one.
- Define an algorithm.
- Algorithm
  - a list of steps to follow to solve the problem in which order is important.
  - specifies the actions to be executed, and the order in which these actions are to be executed.

Rise and Shine Algorithm

- Get out of bed
- Take a shower
- Get dressed
- Eat breakfast
- Drive to work

Badly Ordered

- Get out of bed
- Get dressed
- Take a shower
- Eat breakfast
- Drive to work
  - Arrive in work wet!

Solution Generation Techniques

- Brainstorming
  - Experience helps
- Inversion
  - Working from the bottom to the top.
- Searching the web!
  - Many solutions already exist and can be found if you search for it.
    - Careful: Are they reliable?
    - Not everything on the web is correct.

Implement the Solution

- Write down the rules.
- Build the machine (hardware).
- Write the program (software).

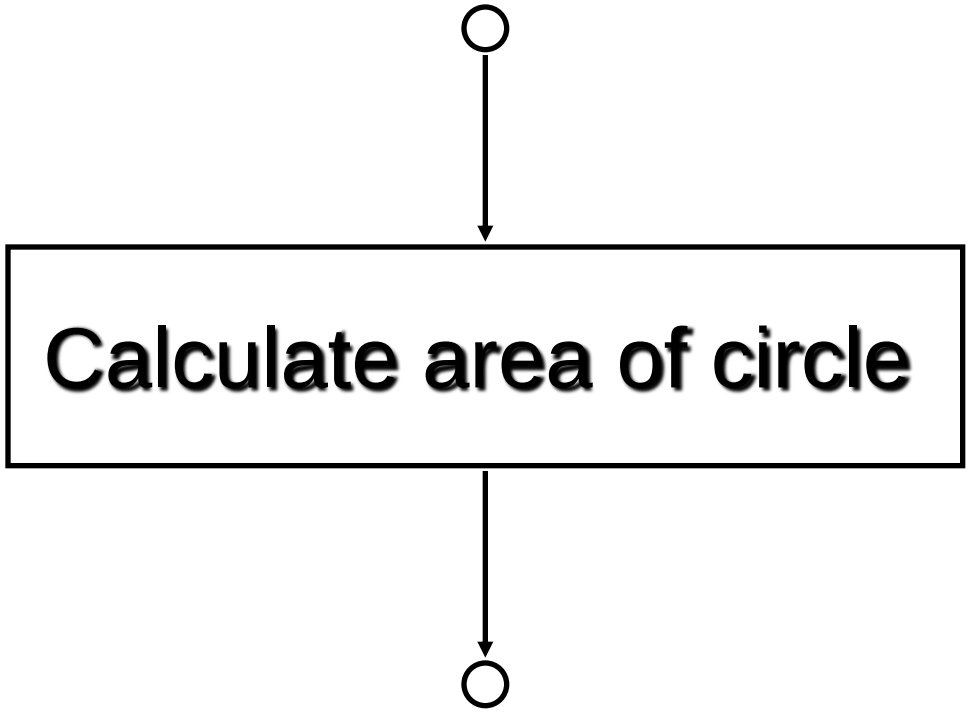
Test the Solution

- Check that the solution works in many different situations.
- Try different inputs.
- Test with the wrong inputs.

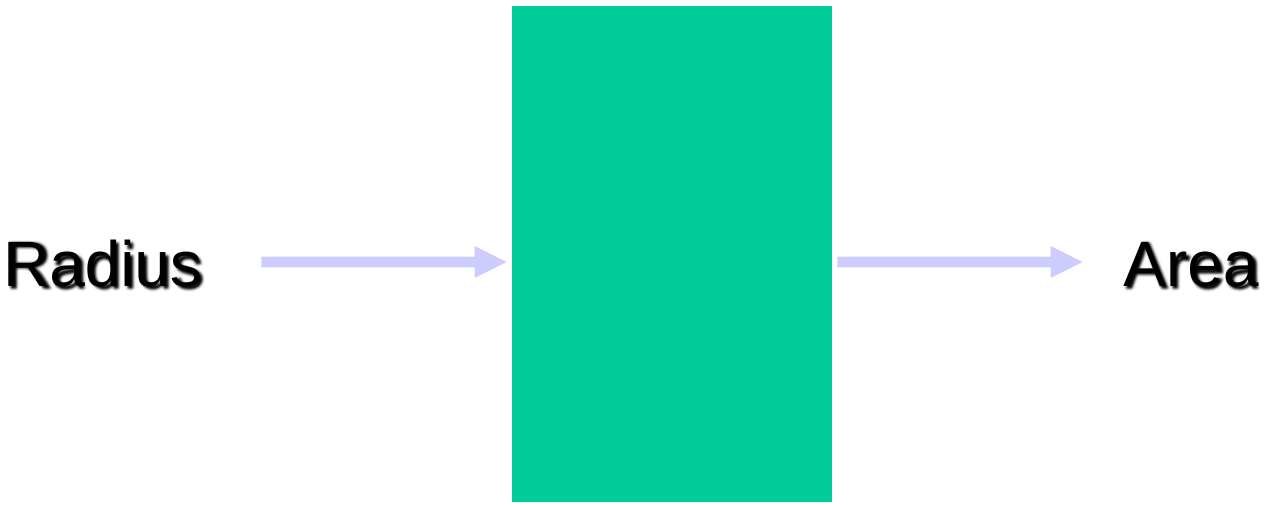
Examples

Problem Definition 1

- How to calculate the area of a circle?



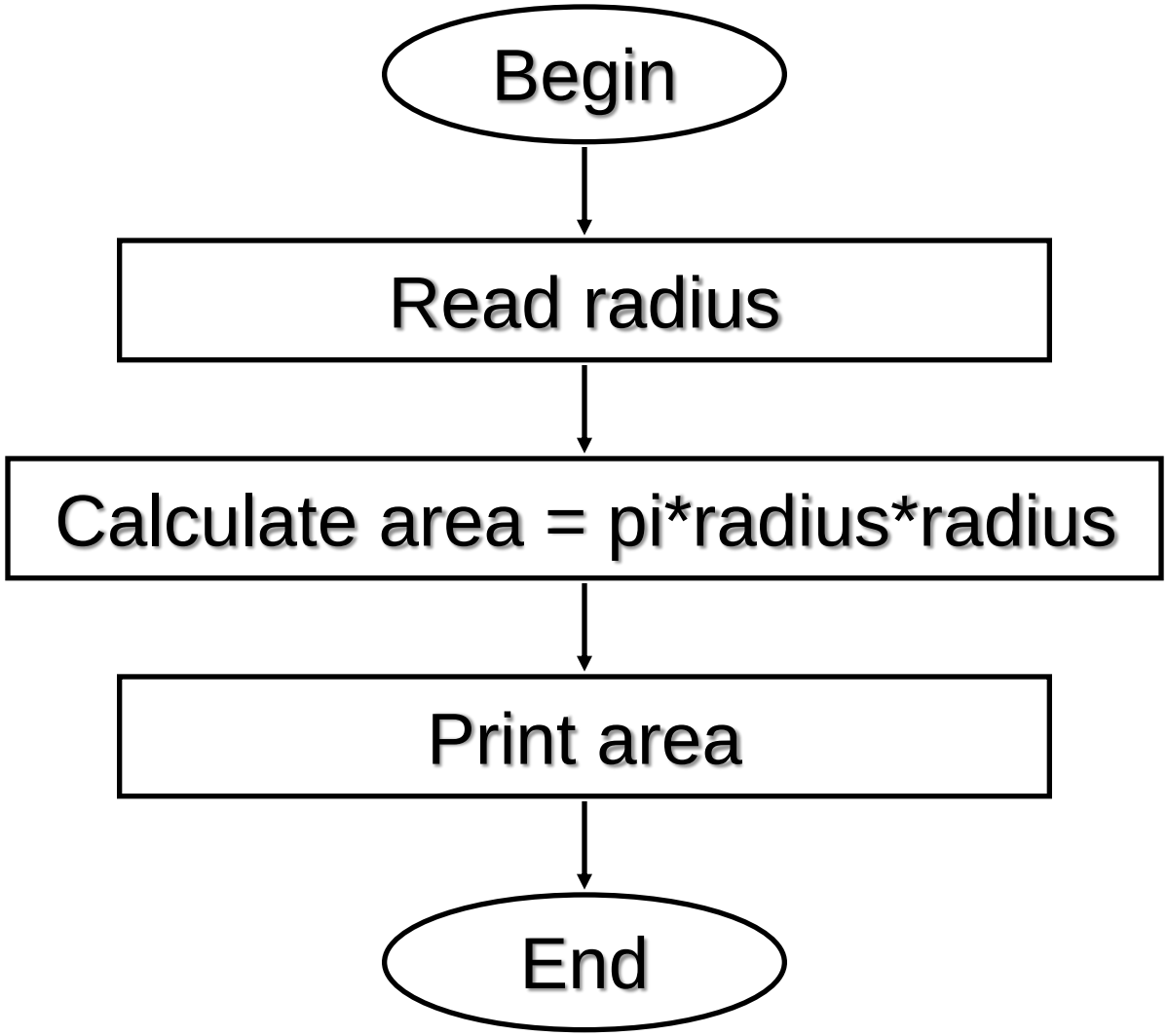
Input-Output Description



Gather Information

- Mathematics: $a = \pi r^2$
- area = π x radius x radius
- area = pi * radius * radius

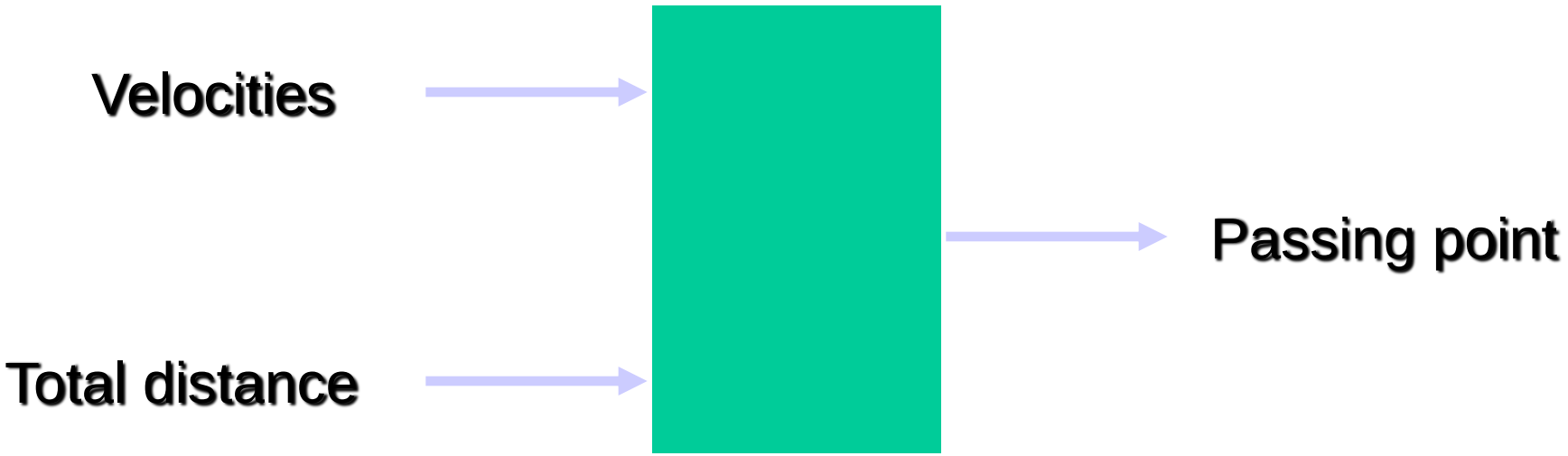
Algorithm


Begin
Read radius
Calculate area = pi*radius*radius
Print area
End

Problem Definition 2

- A train leaves Montreal at 10 AM travelling towards Toronto at 100km/h.
- At the same time a train leaves Toronto going towards Montreal at 150km/h.
- If the distance from Montreal to Toronto is 550km, where do the trains pass each other?

Input-Output Description



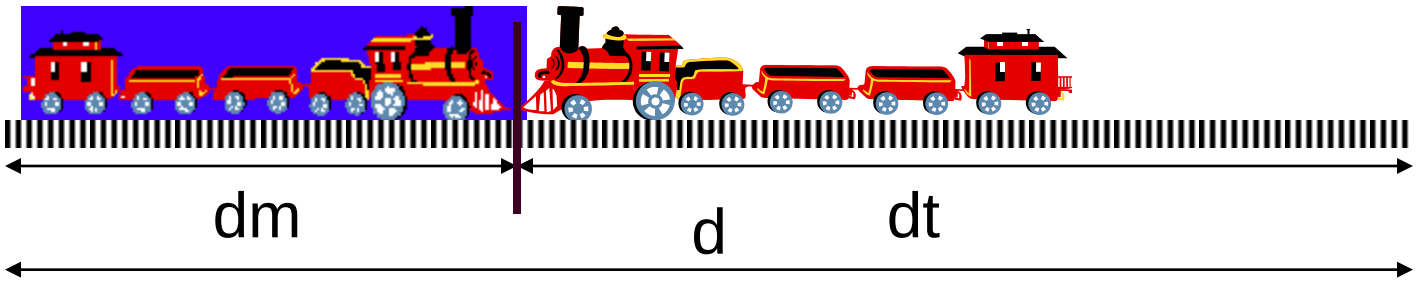
Gather Information (1/2)

- distance = velocity * time
- For the train that leaves Montreal:

  $\mathbf{d_m = v_m * t} \rightarrow dm = vm * t$
- For the train that leaves Toronto:

  $dt = vt * t$
- The total distance Montreal to Toronto is:

  $d = dm + dt = 550$ Km



Gather Information (2/2)

- The trains pass each other after travelling for the same time t (same start time):

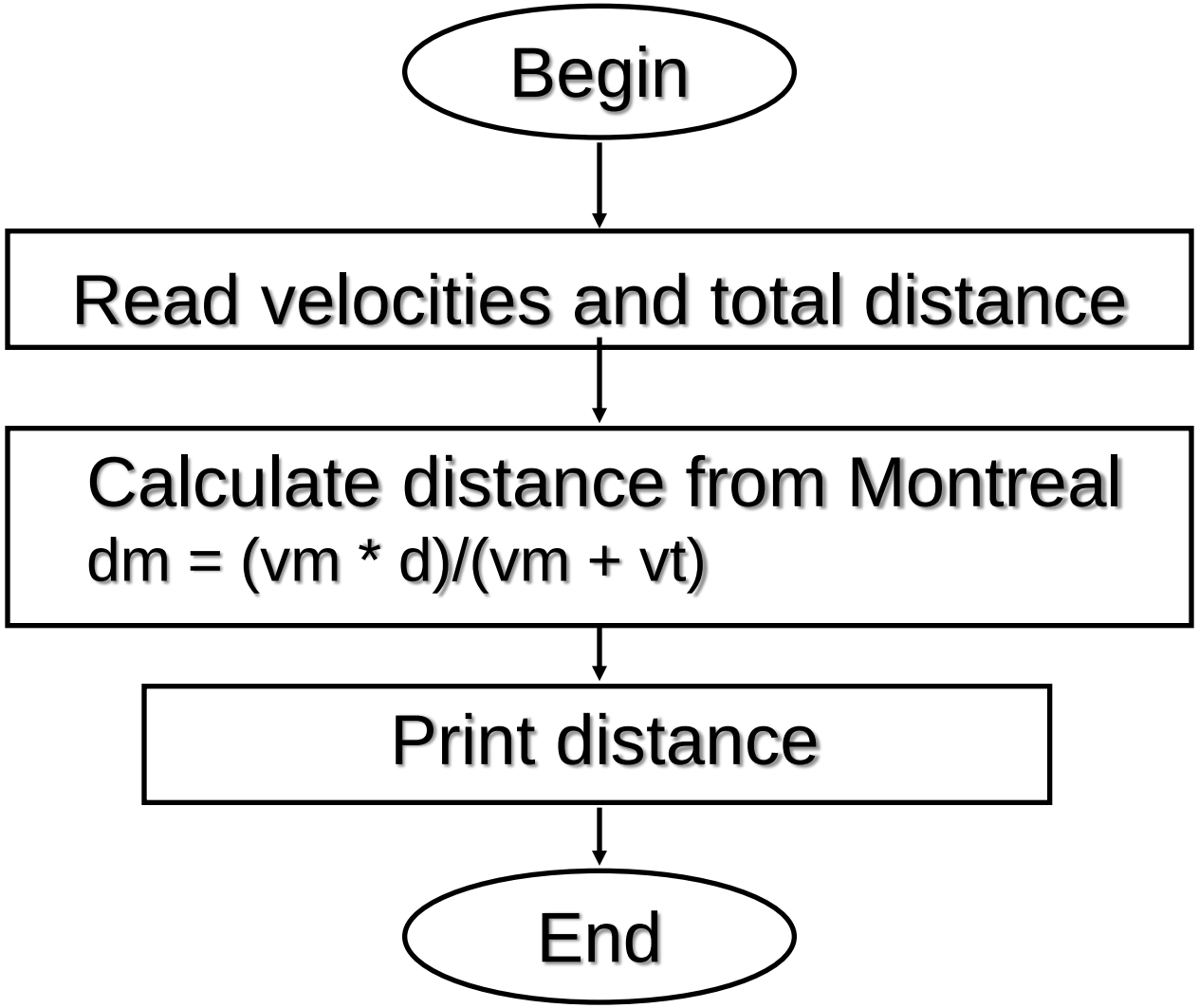
$$d = dm + dt = (vm + vt) * t$$

$$t = d/(vm + vt)$$

$$dm = vm * t$$

$$dm = (vm * d)/(vm + vt)$$

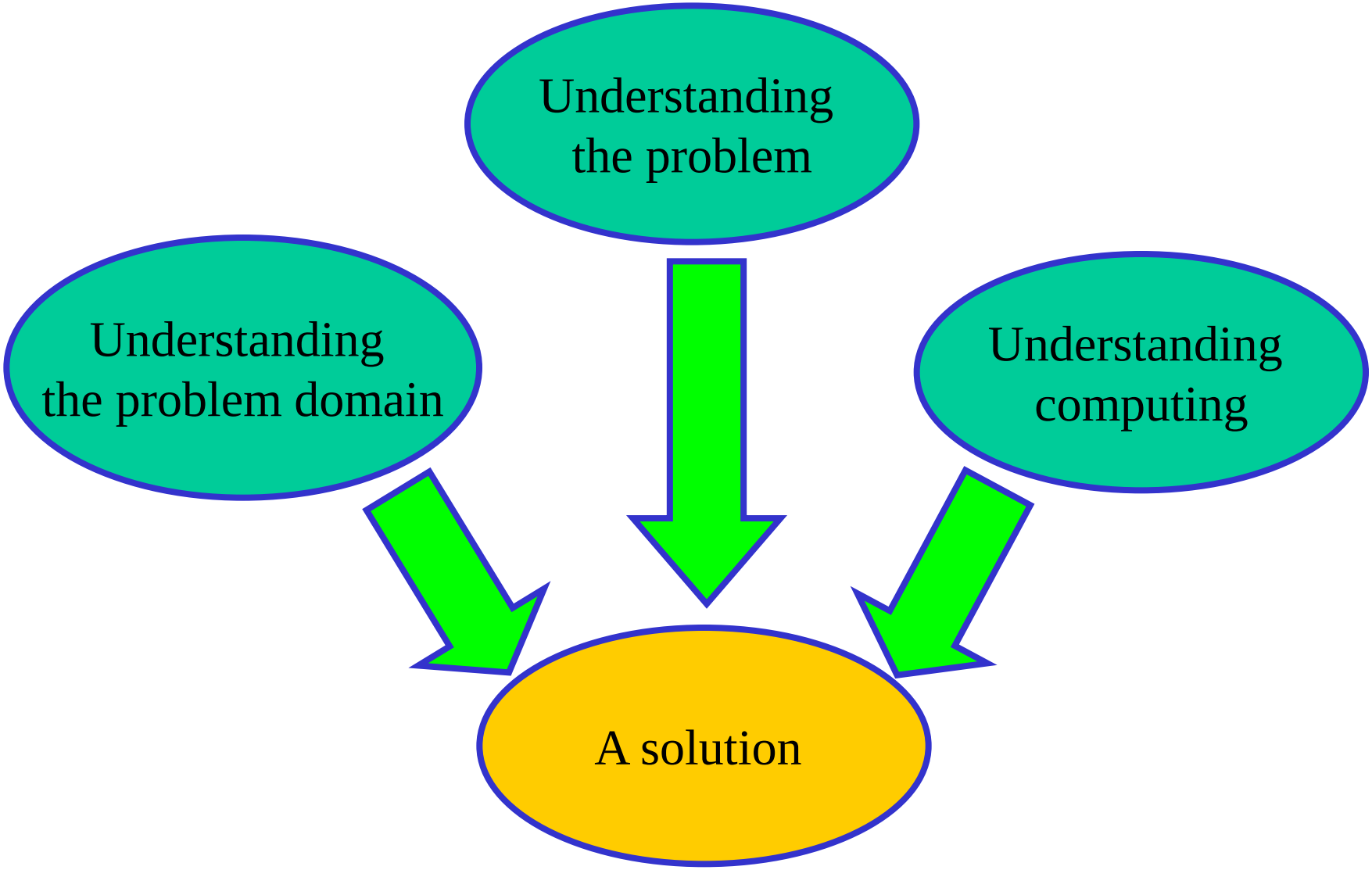
Algorithm


Begin
Read velocities and total distance
Calculate distance from Montreal
dm = (vm * d)/(vm + vt)
Print distance
End

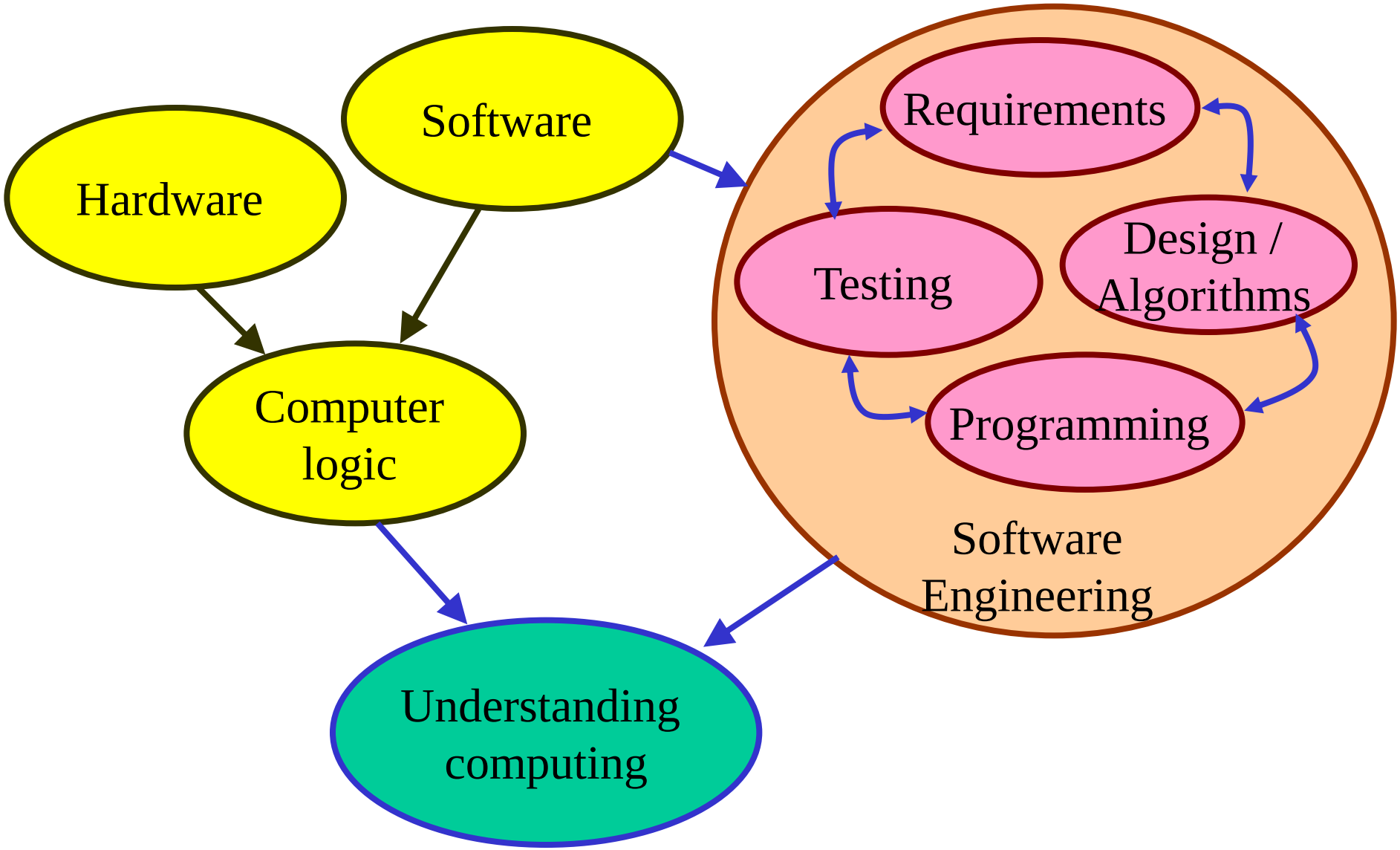
Testing

- dm = (100 * 550)/(100 + 150) = 220km
- dt = d - dm = 550 - 220 = 330km
- t = dm/vm = 220/100 = 2.2h
- dt = vt * t = 150 * 2.2 = 330km  Verification

Solving problems with computing



Some Computing Concepts



Introduction to Computers

- What is a computer?
  - A computer is a device that is capable of making calculations and logical decisions at speeds MUCH FASTER (10^6 to 10^9 times faster) than humans.
    - e.g.: A PC in 1996 could do about 10^7 additions/sec while a super-computer could do about 10^{11} additions/s.
  - A computer operates on data according to directives or commands that are listed in a program. The program is written by a human (usually) using a programming language (This is computation!).
  - Both the computer (hardware device) and the program (software or commands) are required for a computer to accomplish a useful task.

Computer Organization (1/2)

The logical organization of most general purpose computers is as follows.

- **Input Unit.**
  - Obtains information such as data and programs using devices such as a keyboard, diskettes, hard drives and network access cards.
- **Output Unit.**
  - Makes available to the outside world the results of computations using devices such as a screen, speakers, network access, etc.
- **Memory Unit (RAM).**
  - Has low storage capacity compared to a hard drive but is rapidly accessed.
  - Storage is temporary and data is lost when power to the memory unit is turned off.
  - Usually contains the programs in execution as well as the required input data and some output data.

Computer Organization (2/2)

- **Central Processing Unit (CPU).**
  - Administrative section of the computer which coordinates the operation of the other sections. Synchronizes the operation of the computer.
- **Arithmetic Logic Unit (ALU).**
  - Contains computing mechanisms to perform basic arithmetic operations such as addition, subtraction, multiplication and division, as well as the mechanisms required for logical decision making.
  - It is part of the CPU.
- **Secondary Storage Unit.**
  - Long term, usually high capacity storage unit, such as hard drive.
  - Access is slow relative to the memory unit.
  - Contains programs and data that are not in immediate use.

What is Computation?

- The processing of data according to an algorithm – or problem solving!
- **Knowledge is divided into two categories**
  - Declarative
    - ❖ Statements of facts!
    - ❖ What are things!
  - Imperative
    - ❖ A description of how to do something (it is a recipe).
    - ❖This is computation!

  - ➢ Example: Square root!

Programming Language

- **To describe algorithms, we need a language.**
  - C programming Language is used in this course.

- **Syntax**
  - What are the legal expressions in the language.
  - Example: She, he, it.

- **Semantics**
  - Which expressions make sense.
  - What does the program mean (what happens when I run a program)
  - Example of incorrect semantics (even though syntax is correct)
    - My desk is Alex!

Programming Language Types (1/2)

Three classifications or categories of programming languages.

- **Machine code**
  - The only language that the computer hardware can understand directly.
  - All programs and data must be translated to machine language.
  - Language is machine dependant.
  - Machine code is generally difficult to manipulate since it consists essentially of numbers.
    - e.g.: Machine code for adding variables x and y and storing the result in z:

```
FC 0900
F3 0902
7C 0904
```

Programming Language Types (2/2)

- **Assembly Language**
  - Machine code's numerical commands are associated to English-like abbreviations or mnemonics describing the command.
  - A program is written using the abbreviations and an **assembler** translates the program into machine code. Assembly language is still too cumbersome to manipulate for writing large and complex programs.
    - e.g.: Assembly code for adding two variables x and y and storing the result in z:
      - LOAD x
      - ADD y
      - STORE z

- **High Level Languages**
  - Consist of a large number of powerful commands.
  - Commands are descriptive, mathematical and/or English-like.
  - A **compiler** translates the program written in high level language to a program in machine code.
    - e.g.: C code for adding two variables x and y and storing the result in z:

      z = x + y;

Programming lifecycle

- The programming or coding process, using a high level language like C, involves many iterations of the following sequence:
  - **Editing** of the program using a text editor or word processor. An editor allows the programmer to write his/her code into electronic form and to save it in a computer file having a filename of the form: `program1.c`
  - **Preprocessing** executes the preprocessor directives that apply to the program code before compilation. Common preprocessor directives include substitution and insertion of code.
  - **Compilation** of the code using the required language compiler. This translates the high level language program written by the programmer to machine language. A machine language file is created having the form: `program1.obj`
  - **Linking** the object code with the necessary libraries (mathematical, input/output and character string manipulation). The linker creates an executable file that can be loaded on the machine; the executable filename generally has the form: `program1.exe`
  - **Loading** of the executable code into memory.
  - **Execution** of the program by the computer.

Popular High-Level Languages

1. COBOL (COmmon Business Oriented Language)
2. FORTRAN (FORmula TRANslation)
3. BASIC (Beginner All-purpose Symbolic Instructional Code)
4. Pascal (named for Blaise Pascal)
5. Ada (named for Ada Lovelace)
6. Java (very commonly used)
7. Visual Basic (Basic-like visual language developed by Microsoft)
8. Delphi (Pascal-like visual language developed by Borland)
9. C++ (an object-oriented language, based on C)
10. C# (very similar to Java, created by Microsfot)
11. C (We use it in this course)

Pseudo-code

- Pseudo-code instructions consists of English language text to describe the desired action to be taken by the computer. e.g.

  Assign 2 to var1

- A single line of pseudo-code normally corresponds to a single programming language instruction.
  - Sometimes a single line of pseudo-code corresponds to multiple programming language instructions.
- Instruction blocs (a sequence of instructions) are important when creating coding structures (decision and looping structures)
  - Instructions in such blocs have the same indentation in pseudo-code (see later).
- In course notes, pseudo-code is written in the ***Script*** font.