Problem Understanding and Analysis

What is a problem?

- There are many different kinds of problems:
- How much tax should my employees be paying?
- How do I get to Oxford from London?
- Why is it wrong to lie?

Cont..

- However, not all kinds of problems can be solved using computers (Which of the above do you think can and cannot be solved with a computer?).

- For a problem to be solvable through the use of a computer, it must generally:

  - Be technical in nature with **objective** answers that can be arrived at using **rational** methods
  - Be **well-structured**
  - Contain sufficient information for a solution to be found
  - Contain little, if any, **ambiguity**.

How do we solve problems?

- We need to THINK!

- We need to engage in one or more of the following types of thinking:

  - **Logical reasoning** (e.g. *Dave and Anne have two daughters. Each daughter has a brother. How many people are there in the family?).*

  - **Mathematical reasoning** (e.g. *A car has a fuel capacity of 10 gallons and a fuel consumption of 45 miles per gallon. How far can the car travel on three quarters of a tank?).*

  - **Lateral thinking** (e.g. *I throw a heavy weight to arrive at a district in North London?).*

How do we solve problems?

- Problem solving is easier if we employ a **problem solving framework** and an appropriate **problem solving strategy to** aid us.

A problem solving framework

- An outline framework for problem solving:
  1. Understand the problem
  2. Devise a plan to solve the problem
  3. Carry out the plan
  4. Assess the result
  5. Describe what has been learned from the process
  6. Document the solution.
- This framework is called **How to Think Like a Programmer** (HTTLAP) approach to problem solving.

(1) Understanding the problem

- To understand a problem
  - We need to read and reread it till we understand every detail
  - We need to dissect the problem into its component parts (e.g. problems and **sub-problems**)
  - We need to remove any **ambiguity**
  - We need to remove any information that is **extraneous** to the problem
  - We need to determine our **knowns** and our **unknowns**
  - We need to be aware of any **assumptions** we are making.

(2) Devise a plan to the solve the problem

- If a problem contains a set of sub-problems, in what order are you going to solve them?

- How are you going to represent the problem:
  - Numerically?
  - Graphically?
  - Tabular data?
  - Natural language?

- Does the problem lend itself to a particular problem solving strategy or strategies:
  - Working backwards?
  - Solving a simpler analogous problem?
  - Logical reasoning?
  - Finding a pattern?

(3) Carry out the plan

- Consider the following problem:

  In a room with ten people, everyone shakes hands with everyone else exactly once. In total, how many handshakes are there?
- How would you represent and solve the problem?
- To solve it quickly and correctly without guessing we could use one of the strategies mentioned earlier. However, which strategy is best suited to solving this problem?
- To answer this, we need to familiarize ourselves with the strategies and practice their use.

Cont..

- To answer the handshakes problem we could begin by **accounting for all the possibilities** and graphically represent those possibilities in a table.
- This clearly shows us who shakes hand with who (A to J shake hands with everybody except themselves (X)).
- We can then use some basic **mathematical reasoning** to find our solution:

  number of people = 10

  grid size = 10 x 10 = 100

  handshakes = (100 - 10)/2 = 45

- From here we can abstract a generalisable solution that can easily be turned into an algorithm and eventually into computer code:

  $h = ((p^2) – p))/2$

	A	B	C	D	E	F	G	H	I	J
A	X									
B		X								
C			X							
D				X						
E					X					
F						X				
G							X			
H								X		
I									X	
J										X

(4) Assessing the results

- It is very unusual when solving complex problems to achieve the correct result first time round. We often need several attempts to get it right.
- To verify our solutions are correct, we need to take a few steps backwards:
  - Was our understanding of the problem correct?
  - Did we overlook anything?
  - Did we choose the correct strategy?
  - Did we employ that strategy correctly?
  - Have we made any incorrect or unwitting assumptions?
- However, it is often very difficult to spot our own mistakes. It is often better, therefore, to have somebody else verify our solutions for us.

Cont…

- Sometimes solutions appear correct, but are in fact wrong, due to an initial misunderstanding of a problem
- If you have misunderstood a problem, it does not matter how good a coder you are, your program will not work as it is supposed to.
- Therefore getting the problem-solving part of programming right is absolutely essential if we are to build programs that work as they are supposed to work.

(5) Describing what you have learned

- You can only become a good problem solver by reflecting on your experiences of problem solving.

- Keeping a record of problems you have attempted, your success, failures, the approaches you have used, etc. will:
  - Broaden your problem solving repertoire
  - Help you to recognize similarities/patterns in problems
  - Help you identify and fix logical or implementation errors
  - Help you to solve problems faster and more effectively

(6) Documenting the solution

- Documenting a solution will help you to **generalize** your approach to other similar problems.
- Do you recognize the following type of problem? How would you solve it?
- It will also prevent you forgetting how you arrived at your solutions.
- In the long run, it will make you a better problem solver and eventually a better programmer.

Strategies for Problem Solving

Strategies for arriving at solutions include:

Problem solving refers to the thinking we do in order to answer a complex question or to figure out how to resolve an unfavorable situation.

trial and error

Trial and error involves trying various possible solutions, and if that fails, trying others.

- When it's useful: perfecting an invention like the light bulb by trying a thousand filaments
- When it fails: when there is a clear solution but trial and error might miss it forever

algorithms

An algorithm is a step by step strategy for solving a problem, methodically leading to a specific solution.

heuristics

A heuristic is a short-cut, step-saving thinking strategy or principle which generates a solution quickly (but possibly in error).

insight

Insight refers to a sudden realization, a leap forward in thinking, that leads to a solution.

Clarifying Problem Solving Examples

Where's the apple juice? Do I look on every shelf in the store, or do I search where there is similar stuff?

Trial and error — Wander around a supermarket randomly to find it.

Algorithms — Create a methodical path to make sure you check every single aisle.

Heuristics — Check only related aisles.

End of Lecture 9