# Lesson 2 Algorithms for everyday tasks

## Learning goals

1. Give examples of computing tasks that arise in
   1. everyday life (e.g. finding winning move in tic-tac-toe)
   2. math and science classes (e.g. solving a linear system)
2. Design algorithms for familiar computing tasks and implement them in pseudocode using
   1. sequential structures
   2. decision structures
   3. loops
3. Develop analytical skills that will be needed in later units when we learn computer programming.

## Agenda

1. Thanks for the letters in HW #1-1.
2. Binder check + reminder of how we start class – 2 minutes
3. Algorithms you already know – 10 min
4. Designing algorithms in pseudocode – 40 min
   1. sequential structures (wolf sheep cabbage)
   2. conditional structures (color mixing)
   3. loops (change counting)
5. Kids watch “The Four Types of Computer Scientists.mov”
6. Begin HW #1-2 – 20 min

### Algorithms you already know

Word of the day: *algorithm*

|  |  |
| --- | --- |
| What does each group have in common? | Answer |
| Add 7 + 7 + 7 + 7 + 7  Count 5, 10, 15, 20, 25, 30, 35  Post on Facebook “Help! What’s 7 x 5?‼” | These are all *algorithms* for  multiplying 5 X 7  *Are they all equally efficient?* |
| Eenie meenie miney mo  Draw straws  Hunger Games tournament | These are all *algorithms* for picking a winner from a group of people |
| Substitution  Elimination  Graphing  Pick random (x,y) until you find a pair that works. | These are all *algorithms* for task of  solving a linear system  *Are they all efficient? Guaranteed to work?* |

Moral:

There can be many algorithms for doing a task…but not all of them are efficient or even guaranteed to work in all cases.

### Designing algorithms in pseudocode

We can view a *task* as an *input* with a desired *output*.

An *algorithm* for that task is any procedure that generates the output from the input.

**Task 1:** Solve the Wolf-Sheep-Cabbage problem

|  |  |
| --- | --- |
| Input: A farmer, a wolf, a sheep, a cabbage and a boat on one bank of a river (side A) | Output: All objects safely on the opposite bank (side B) |
| Algorithm:  Load sheep  Sail to B  Unload  Sail to A  Load wolf  Sail to B  Unload  Load the sheep 🡪 | Sail to A  Unload  Load cabbage  Sail to B  Unload  Sail to A  Load sheep  Sail to B  Unload |

**Task 2** Start a car

|  |  |
| --- | --- |
| **Input**: Car owner standing outside locked car with key in pocket | **Output**: Owner in driver’s seat with car running and in gear |
| **Algorithm**  Get key from pocket\*  Unlock car\*  Get into car\*  If key type = mechanical:  Insert key into ignition  Depress brake  Turn key and hold until engine starts  Release key  \* each of these requires an algorithm of its own | If key type = transponder:  Depress brake  Push Start button  If desired direction = forward:  Move gear shifter to D  Else if desired direction = reverse:  Move gear shifter to R  Else:  Leave gear shifter in P |

Do you think someone could write a computer program to make a robot follow this algorithm for starting a car?

**Task 3** Set the table for 10 guests

|  |  |
| --- | --- |
| Input:  An empty table with dishes stacked in kitchen, and robot in kitchen | Output:  A table set for 10 people |
| Algorithm:  Gather 10 plates, forks, spoons, knives and glasses  Go to your own chair  Do 10 times:  Set plate  Set fork to left of plate  Set knife and spoon to right of plate  Set glass to upper left of plate  Move one place clockwise  Ring dinner bell | |

Watch “The Four Types of Computer Scientists.mov” at the computers. Open it as a QuickTime movie and click through the slides.

Then start HW #1-2