A High-Level Introduction to Algorithms

Lecturer:

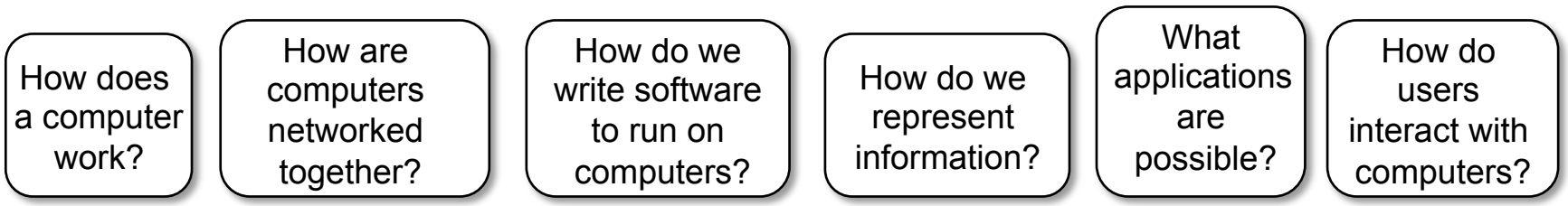
Professor Barry O'Sullivan

Office: 2-65, Western Gateway Building

email: *b.osullivan@cs.ucc.ie*

http://osullivan.ucc.ie/teaching/cs1112/

Overview of Computer Science



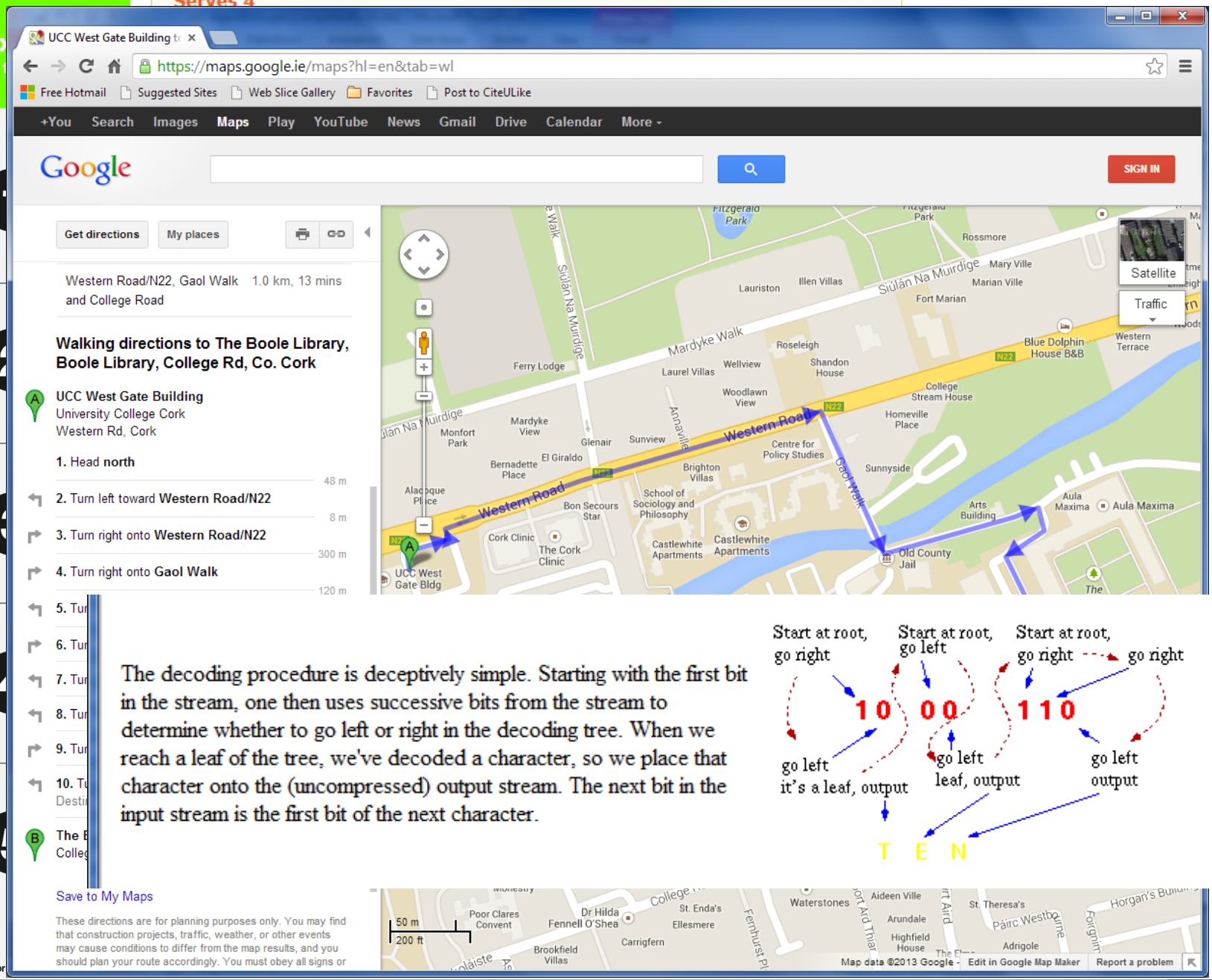
The Fundamental Concept of Computing

- Anything a computer does is based on carrying out a sequence of steps. E.g.
  - Programs
  - Low level operations
  - Translating high-level programs to low-level steps
  - Exchanging information between one computer and another anywhere in the world
  - Reading an html file and turning it into a visible web page
  - Examining each email you receive to check for viruses
  - Accepting your commands and converting them into actions in a game on a console
  - ... and many more

- A sequence of steps to achieve a task is an ***algorithm***

Algorithms

- An algorithm is a set of instructions which state how a task is to be performed


UCC West Gate Building
https://maps.google.ie/maps?hl=en&tab=wl
Free Hotmail
Suggested Sites
Web Slice Gallery
Favorites
Post to CiteULike
+You
Search
Images
Maps
Play
YouTube
News
Gmail
Drive
Calendar
More
Google
SIGN IN
Get directions
My places
Western Road/N22, Gaol Walk 1.0 km, 13 mins
and College Road
Walking directions to The Boole Library, Boole Library, College Rd, Co. Cork
UCC West Gate Building
University College Cork
Western Rd, Cork
1. Head north
48 m
2. Turn left toward Western Road/N22
8 m
3. Turn right onto Western Road/N22
300 m
4. Turn right onto Gaol Walk
120 m
Save to My Maps
These directions are for planning purposes only. You may find that construction projects, traffic, weather, or other events may cause conditions to differ from the map results, and you should plan your route accordingly. You must obey all signs or
Satellite
Traffic
Western Road
Mardyke Walk
Siúlán Na Muirdige
College Stream House
Blue Dolphin House B&B
Old County Jail
Arts Building
Aula Maxima
Castlewhite Apartments
Cork Clinic
The Cork Clinic
UCC West Gate Bldg
Map data ©2013 Google
Edit in Google Map Maker
Report a problem
The decoding procedure is deceptively simple. Starting with the first bit in the stream, one then uses successive bits from the stream to determine whether to go left or right in the decoding tree. When we reach a leaf of the tree, we've decoded a character, so we place that character onto the (uncompressed) output stream. The next bit in the input stream is the first bit of the next character.
Start at root, go right
Start at root, go left
Start at root, go right
go right
1 0
0 0
1 1 0
go left it's a leaf, output
go left leaf, output
go left output
T E N

Algorithms

- An algorithm is a set of instructions which state how a task is to be performed
  - A knitting pattern
  - Instructions for setting up a new Xbox console
  - A recipe for cooking spaghetti bolognaise
  - Directions for walking from Boole Library to 1st year Computer Science lab in Western Gateway Building
  - The operations for turning an MP3 file into a sequence of sounds produced by an MP3 player

An algorithm is an ordered, deterministic, executable, terminating set of instructions

An algorithm is an ordered, deterministic, executable, terminating set of instructions

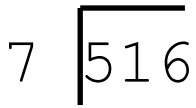
The order is important.
Do you want to do step 3 before 2?

The Rise & Shine Algorithm

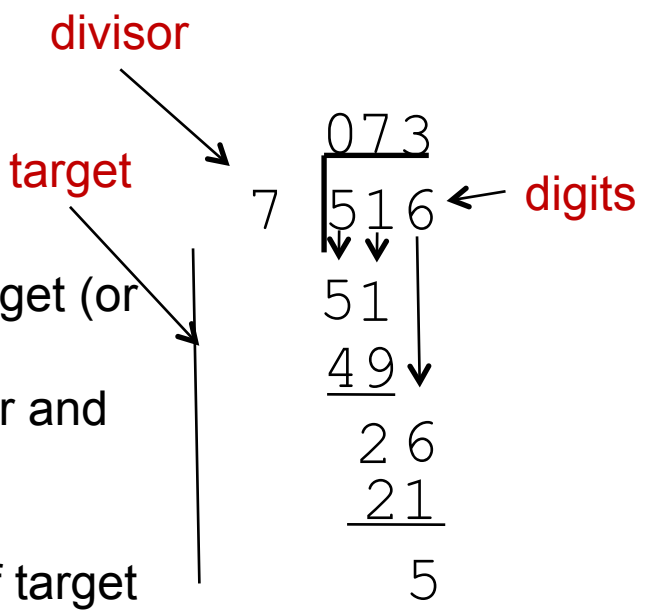
1. Get out of bed
2. Have a shower
3. Put on underwear
4. Put on outer clothes
5. Have breakfast
6. Come in to UCC

... Or step 4 before 3?

How would you do this division?



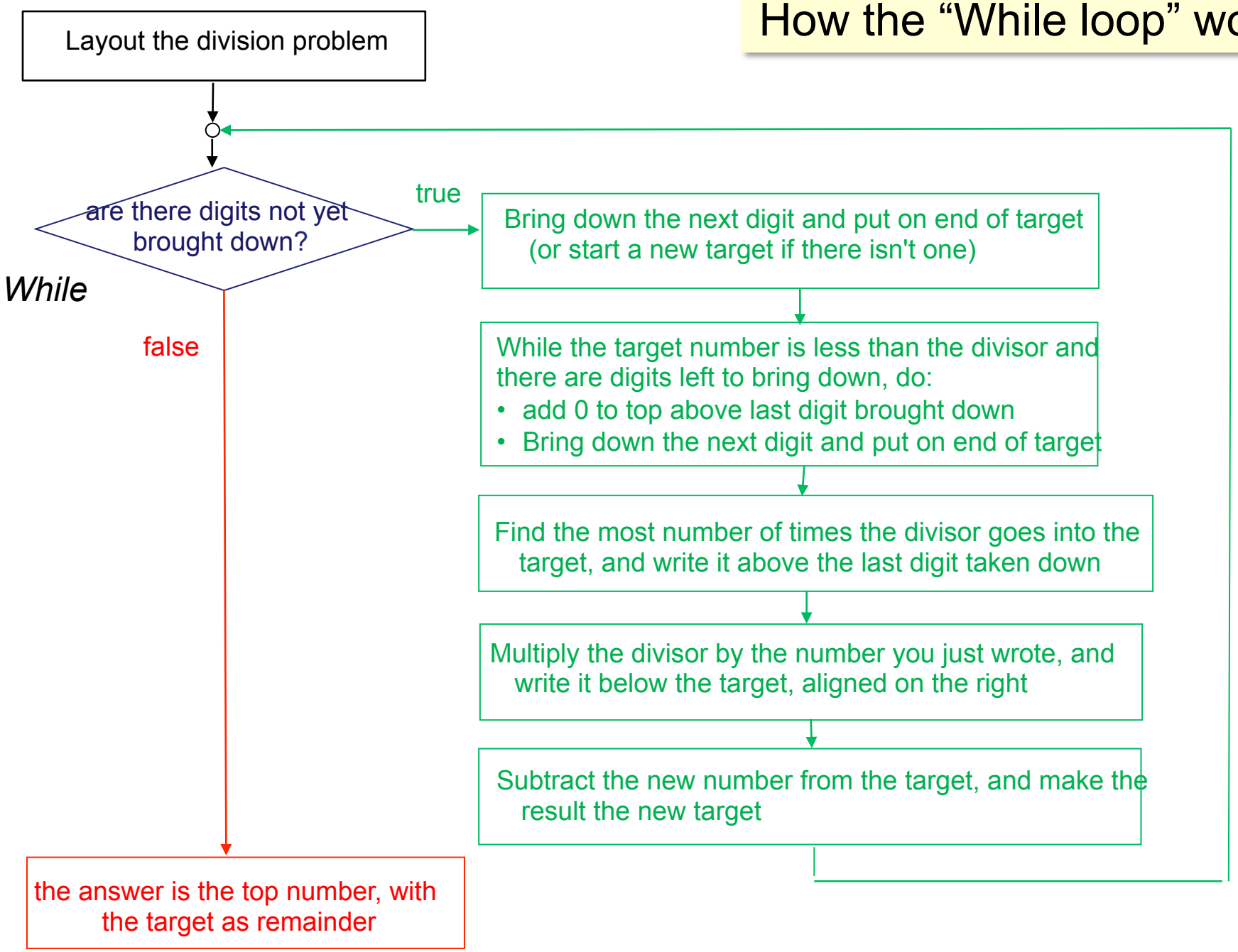
"ordered" does not mean just a simple sequence



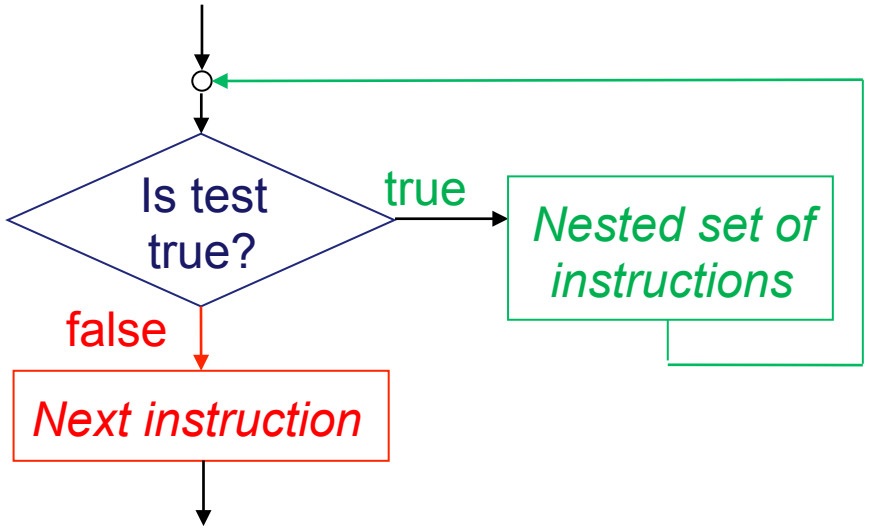
- Layout the division problem,
- While there are digits not yet brought down, do:
  - Bring down the next digit and put on end of target (or start a new target if there isn't one)
  - While the target number is less than the divisor and there are digits left to bring down, do:
    - add 0 to top above last digit brought down
    - Bring down the next digit and put on end of target
  - Find the most number of times the divisor goes into the target, and write it above the last digit taken down
  - Multiply the divisor by the number you just wrote, and write it below the target, aligned on the right
  - Subtract the new number from the target, and make the result the new target
- When there are no digits left to bring down, the answer is then the top number, with the target as remainder

```
516/7 = 73 remainder 5
```

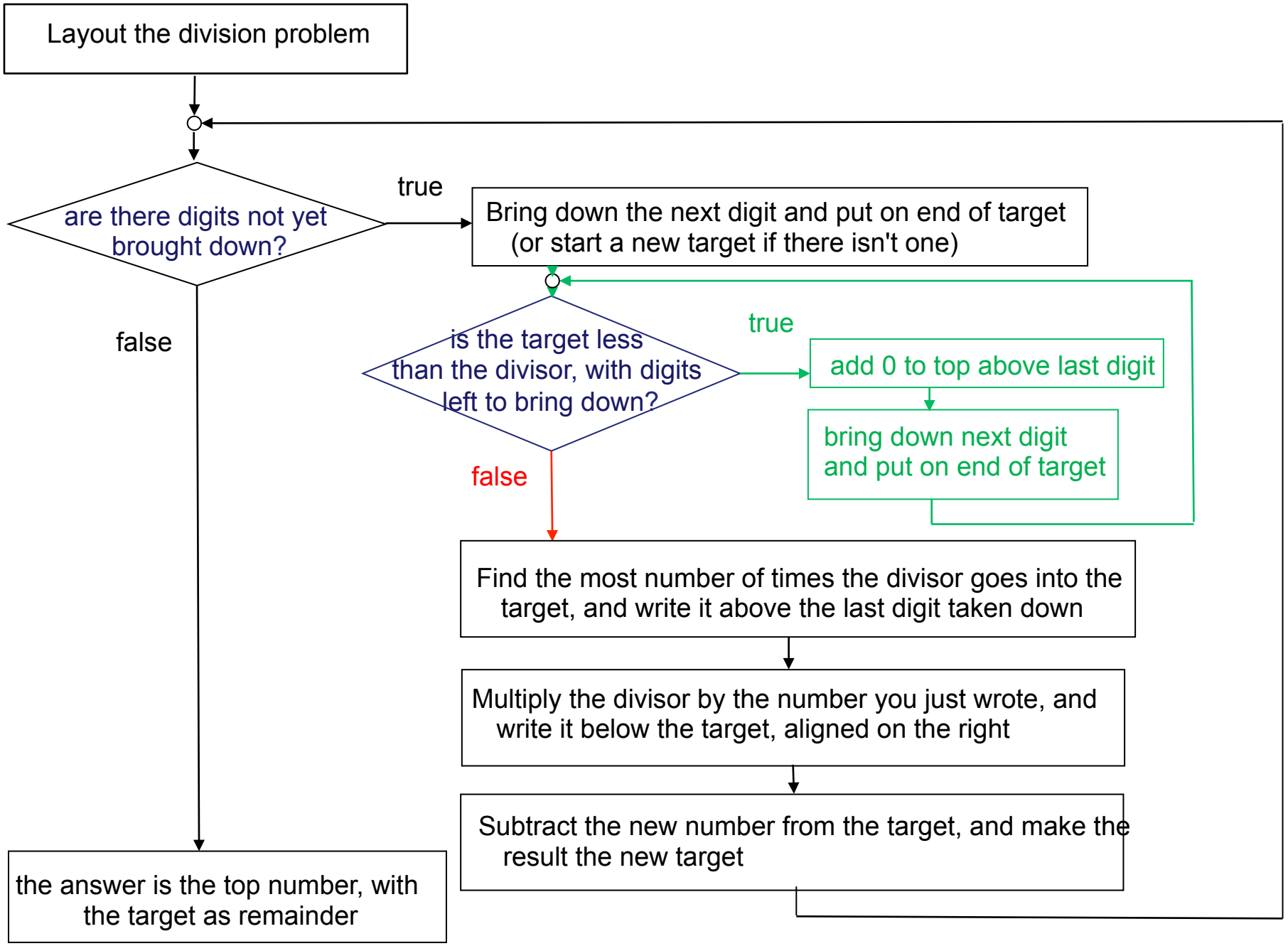
How the “While loop” works



How the “While loop” works



- Layout the division problem
- While there are digits not yet brought down, do:
  - Bring down the next digit and put on end of target (or start a new target if there isn't one)
  - While the target number is less than the divisor and there are digits left to bring down, do:
    - add 0 to top above last digit brought down
    - Bring down the next digit and put on end of target
  - Find the most number of times the divisor goes into the target, and write it above the last digit taken down
  - Multiply the divisor by the number you just wrote, and write it below the target
  - Subtract the new number from the target, and make the result the new target
- When there are no digits left to bring down, the answer is then the top number, with the target as remainder


Layout the division problem
are there digits not yet brought down?
true
Bring down the next digit and put on end of target (or start a new target if there isn't one)
false
is the target less than the divisor, with digits left to bring down?
true
add 0 to top above last digit
bring down next digit and put on end of target
false
Find the most number of times the divisor goes into the target, and write it above the last digit taken down
Multiply the divisor by the number you just wrote, and write it below the target, aligned on the right
Subtract the new number from the target, and make the result the new target
the answer is the top number, with the target as remainder

Al-Khwarizmi

- 780-850
- Member of the House of Wisdom, Baghdad
- Wrote "*The Compendious Book on Calculation by Completion and Balancing*"
- And "*On Calculation with Hindu Numerals*"

Understanding algorithms

Which of the following is the correct algorithm for a bus driver taking fares from a queue of people?

A

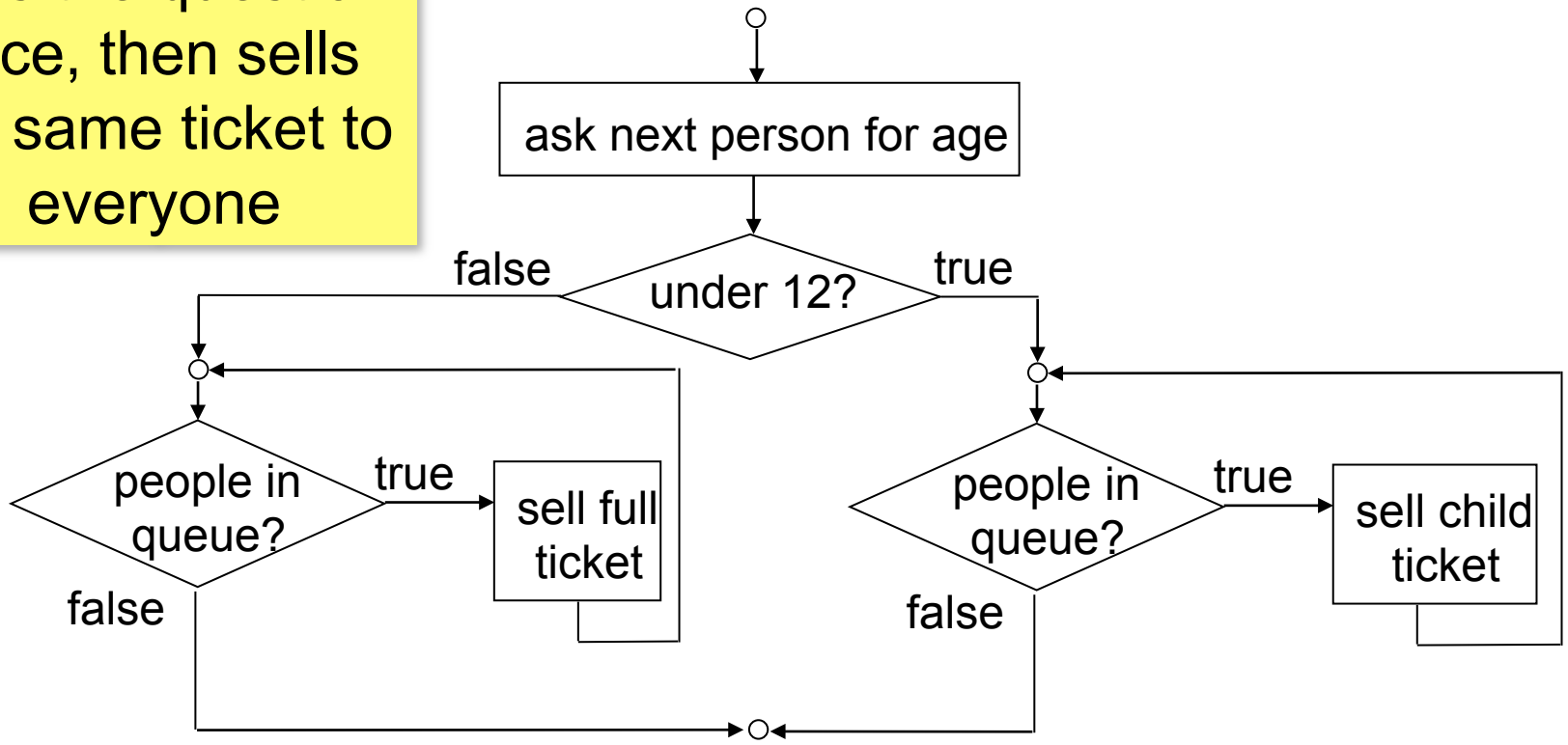
```
ask next person their age
if age less than 12
    while people in queue
*       sell child ticket
else
    while people in queue
*       sell full ticket
```

B

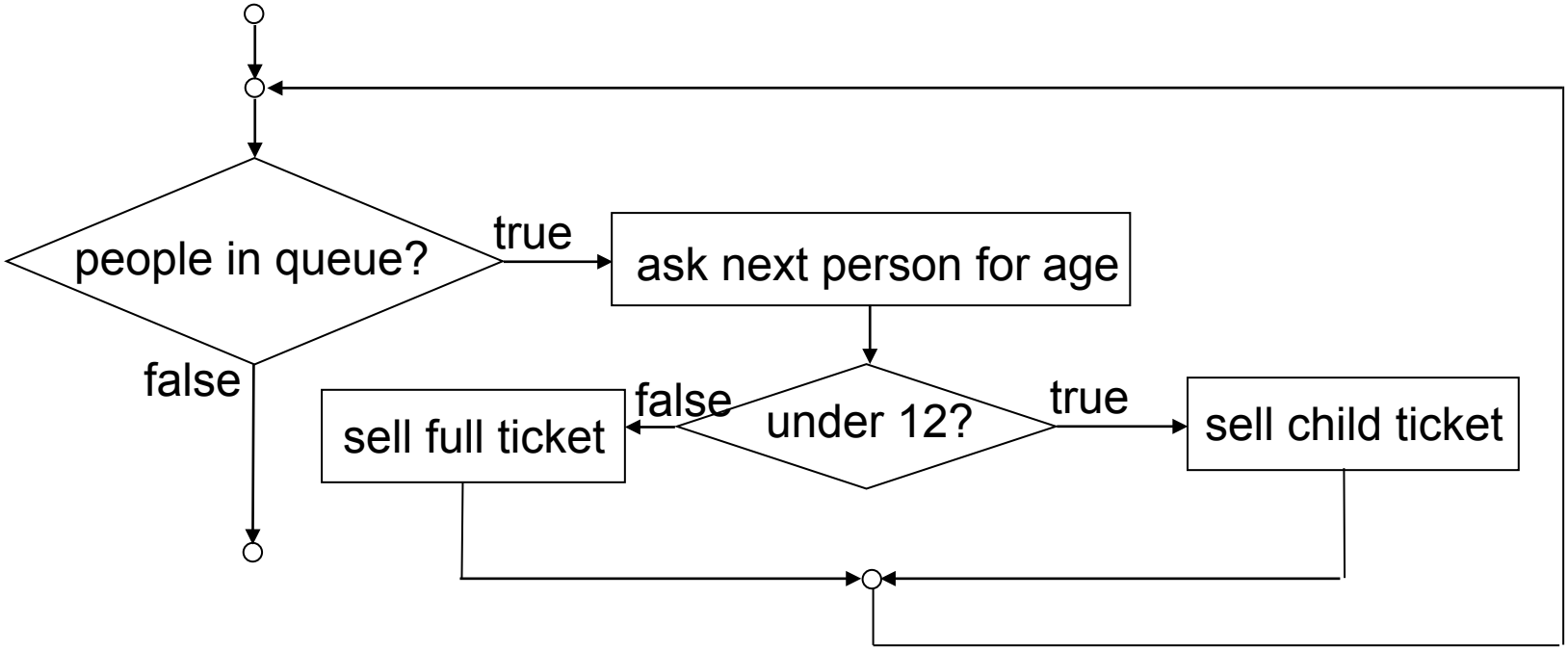
```
while people in queue
    ask next person their age
    if age less than 12
        sell child ticket
*   else
        sell full ticket
```

Bus ticket flowcharts: A

This is wrong - it asks the question once, then sells the same ticket to everyone



Bus ticket flowcharts: B



This is correct - it asks the first person the question, then sells the correct ticket, then moves on to the next person, and asks again.

An algorithm is an ordered, deterministic, executable, terminating set of instructions

Will this set of instructions get you to the airport?

1. Drive along Western Road
2. Drive up to the Sarsfield Roundabout ?
3. Take one of the exits from the roundabout ?
4. Keep driving until you see signs for the airport
5. Follow the airport signs
6. Park car

The instructions must be clear, unambiguous and deterministic for whoever or whatever is going to interpret them

Origami

1. Fold the paper in half along its length, crease the fold, and open it up
2. Fold the top right corner to the centre line to make a right angled triangle, and fold the top left corner in the same way.
3. Fold each side into the centre line.
4. Fold the top of the diamond shape over to the centre line so that the crease is formed at the bottom of the diamond shape, crease the fold, then unfold it. Fold in the opposite direction, crease hard, and unfold.
5. Fold the top right corner to meet the left hand edge, so that the crease runs along the edge of the diamond shape, crease the fold, and then unfold it. Do the same with the top left corner.
6. Put a finger behind each half of the crease you made in step 4, put a thumb on each side of the flat paper below the creases, bring the two edges together, flatten them down in the middle of the paper, and then flatten down the bit that is sticking up, so that you get another diamond shape. Crease the folds to make it lie flat.
7. Fold the bottom half of the diamond up to the top point, crease, unfold.
8. Fold that flap in the opposite direction, and tuck it into the triangular pocket, and crease the fold.
9. Fold the lower right hand flap over in a diagonal line from the bottom corner to just below the triangular shape. Repeat for left hand side.
10. Turn the paper over and fold it in half along its length. Crease lightly.

The instructions must be clear, unambiguous and deterministic for whoever or whatever is going to interpret them

- The instructions in your algorithm may be clear to you, but are they clear to anyone else?

- A program must be written in some *language* if it is to be executed on a computer. You must:
  - understand the language
  - know how statements will be interpreted
  - express yourself clearly in that language

The computer cannot apply common sense

- The only way to become good at writing algorithms and translating them into programs is to get practice
  - Learn what you are allowed to say in the language
  - Learn how each statement will be interpreted
  - Learn how to think clearly and analyse your programs
  - And practise, practise, practise.

An algorithm is an ordered, deterministic, executable, terminating set of instructions

Will this set of instructions get you to the Boole Library?

1. Drive out of the city along Washington St.
2. Drive in through the main gates of UCC ?
3. Drive up the hill until the Student Centre steps
4. Drive up the steps ?
5. Drive across the pavement
6. Park car

A set of instructions that cannot be executed is not an algorithm

A set of instructions that cannot be executed is not an algorithm

- If your algorithm is going to run on a computer, you must write it in a language that can be understood by the computer
- You must understand what sequences of actions can be executed
  - What information needs to be available before a particular action can be performed?
  - How does that information need to be represented?
- The computer will not read ahead and work out what it should be doing

An algorithm is an ordered, deterministic, executable, terminating set of instructions

- This is part of the formal definition of an algorithm, so that we can analyse them, and work out what it is possible for computer to do.
- But sometimes we will ignore this, since we will want to build programs that run forever – e.g. controlling a set of traffic lights
- In later modules we will also see “algorithms” that are not deterministic ...

Programming is mostly about algorithms

- If you understand that instructions must be
  - Ordered
  - Unambiguous and deterministic
  - Executable
- And if you can write algorithms with these properties, then you are halfway to being able to program

- What you still need to do is learn the language
  - what instructions can the computer interpret?
  - how will the computer interpret these instructions?
  - how do we write complex orders?
  - how do we represent information that the algorithms use?

The rest of CS1112?

- How do we talk about collections of data, so that we can be precise and unambiguous in our algorithm descriptions?
  - *Sets, relations, functions, logic*
- How do we talk about conditions, tests, complex instructions, etc., so that we can specify tasks for algorithms, and test that they work?
  - *Sets, logic*

Exercise: Try this for Monday

Origami

1. Fold the paper in half along its length, crease the fold, and open it up
2. Fold the top right corner to the centre line to make a right angled triangle, and fold the top left corner in the same way.
3. Fold each side into the centre line.
4. Fold the top of the diamond shape over to the centre line so that the crease is formed at the bottom of the diamond shape, crease the fold, then unfold it. Fold in the opposite direction, crease hard, and unfold.
5. Fold the top right corner to meet the left hand edge, so that the crease runs along the edge of the diamond shape, crease the fold, and then unfold it. Do the same with the top left corner.
6. Put a finger behind each half of the crease you made in step 4, put a thumb on each side of the flat paper below the creases, bring the two edges together, flatten them down in the middle of the paper, and then flatten down the bit that is sticking up, so that you get another diamond shape. Crease the folds to make it lie flat.
7. Fold the bottom half of the diamond up to the top point, crease, unfold.
8. Fold that flap in the opposite direction, and tuck it into the triangular pocket, and crease the fold.
9. Fold the lower right hand flap over in a diagonal line from the bottom corner to just below the triangular shape. Repeat for left hand side.
10. Turn the paper over and fold it in half along its length. Crease lightly.

Exercise: Try this for Monday

Origami

1. Fold the paper in half along its length, crease the fold, and open it up
2. Fold the top right corner to the centre line to make a right angled triangle, and fold the top left corner in the same way.
3. Fold the top left sloping edge in to the centre line. Repeat on the right.
4. Fold the top tip down to the bottom.
5. Fold the right half of the top edge into the centre line. Repeat on the left.
6. Fold the pointed flap up so that its point meets the top tip.
7. Turn over and fold in half.
8. Fold the top flap over along a line about 2cm from the tip to 4cm down from the top corner. Turn over and repeat on the other side.

Next lecture ...

Representing and reasoning about collections: set theory