# Introduction

Motivation - leitmotif & forcing yourself not to pick example problems that happen to suit the concept being introduced

Problem definition: convert denary number (passed as an integer) in range 0-2100 (a ‘year’ say) to Roman Numerals (returned as a string of characters) using the ‘subtractive’ form (i.e. 4 => IV, not IIII).

# Procedural Programming

## Using a series of While loops

## Extracting re-usable function

Passing by reference - ugh! – but AQA likes it!

# Data structures

General point: can reduce code by making more use of data

## Using arrays

## Using a dictionary

# Recursion

# Functional Programming

Definitions

Techniques: recursion, conditional functions, functional lists

## In C# & VB

## In Haskell

# Assembly Language

Use ARMlite - extends AQA

## Using only Immediate or Direct addressing

Per AQA

## With indexed addressing

Equivalent to array approach

# Automata

## Mealy machines

Write by hand for 0-20, say - as diagrams

Key insight that once you start writing, you write first symbol and transition to the state representing the remainder

Introduce a textual format

Then write a program in high-level language to produce the full thing

## Turing machines

Run on online simulator

1. A Mealy machine in disguise - no real benefit, just a change in format, but provided as a file. Need to use the tape as memory
2. 500 transition functions. Writes one digit at a time, but changes mode for powers of 10
3. Most efficient yet. Converts all digits singly, then converts them according to position; hard part is removing separators

# Logic gates & Boolean algebra

Common idea of the output representation

## Convert 10 input lines to Roman

Can all be done with OR gates, but look for re-use - not just separate circuits for each output pin.

## Convert binary single digit to Roman

Use AND, OR and NOT gates.

Use simplification rules to minimise number of gates

## Expand to full spec, but using BCD input