Section 7.1
An Introduction to Algorithms

Algorithm

- An algorithm is a finite sequence of precise (and effective) instructions for solving a problem
  - effective: capable of being done in a finite amount of time

Algorithm

- Example: Find the largest integer in a finite sequence of integers
  1. Set the temporary maximum to the first integer in the sequence
  2. Compare the next integer in the sequence to the temporary maximum. If it is larger than the temporary maximum, set the temporary maximum to be this integer
  3. Repeat the previous step for the other integers in the sequence
  4. Stop when there are no unexamined integers in the sequence. The temporary maximum is the largest integer in the sequence

Pseudocode

- Pseudocode can be used to describe algorithms
- Pseudocode looks like a real programming language
  - It is precise and unambiguous like a real programming language
  - Its individual instructions are easy to understand for people with programming experience
  - It allows to us to describe algorithms more compactly instead of having to describe them in a natural language such as English

Pseudocode

- Assignment
  - Assigns the result of evaluating an expression to a variable
  - Examples:
    - `x := 4`
    - `z := z + 1`

Pseudocode

- Return
  - Specifies the output of an algorithm
  - Examples:
    - `return(0)`
    - `return(x+y)`

Pseudocode

- If statement
  - Conditional execution
  - Example:
    - ```
      if (a = 0)
          count := count + 1
      end-if
      ```
      ```

Pseudocode

- If-else statement
  - Example:
    - ```
      if (a > 0)
          quotient := b / a;
      else
          quotient := 0;
      end-if
      ```
      ```

Pseudocode

- For loop
  - Repeated execution
  - Example:
    - ```
      for i := 1 to 3
          sum := sum + i
      end-for
      ```
      ```

Pseudocode

- While loop
  - Repeated execution
  - Example: (Count the number of digits in a positive integer)
    - ```
      count := 1
      while (n > 9)
          count := count + 1
          n := n / 10
      end-while
      return(count)
      ```
      ```

Pseudocode

- Nested loops
  - Repeated execution
  - Example: (Count the number of ways that one die roll can be greater than another die roll)
    - ```
      count := 0
      for i := 1 to 6
          for j := i+1 to 6
              count := count + 1
          end-for
      end-for
      return(count)
      ```
      ```

Specifying an Algorithm

An algorithm can be specified by naming it, describing its inputs and outputs, and providing its pseudocode

Remainder Algorithm

- Example: Computing a remainder
  - Name: remainder
  - Input: non-negative integers a and b where $b \neq 0$
  - Output: the remainder after dividing a by b
  - ```
    while (a >= b)
        a := a - b;
    end-while
    return(a)
    ```
    ```

Collatz Sequence

- Example: Computing the next number in the Collatz sequence
  - Name: nextCollatz
  - Input: a positive integer n
  - Output: $n/2$ if n is even; otherwise $3n + 1$
  - ```
    if (remainder(n, 2) = 0)
        n := n / 2
    else
        n := (3 * n) + 1
    end-if
    return(n)
    ```
    ```

Collatz Sequence Length

- Example: Counting the number of steps for the Collatz sequence to converge to 1
  - Name: sequenceLength
  - Input: a positive integer starting point n
  - Output: the number of steps required to get from n to 1
  - ```
    count := 0
    while (n > 1)
        n := nextCollatz(n)
        count := count + 1
    end-while
    return(count)
    ```

(See previous slide)
    ```