

MC215: MATHEMATICAL REASONING
AND DISCRETE STRUCTURES

- Monday, 10/27/08
 - From last time:
 - "Modulo n" equivalence relation
 - Today:
 - Algorithms
 - Pseudocode
- READING: 4.1, Appendix C (Omit 3.5, 3.6)
- EXERCISES:
 - pp. 185-186: 4, 6, 9, 11

Monday, 10/27/08, Slide #1
Printed 10/29/2008 1:33 PM

Chapter 4: Algorithms

- Theorem and Proof: *A fundamental idea of mathematics*
 - Start with a **conjecture**: a claim that some property holds for elements from some set of objects.
 - Design a **proof** that the conjecture is or is not true.
- Problem and Algorithm: *A fundamental idea of computer science*
 - Start with a **problem**: A question or a task concerning elements from some set of objects (the **input**)
 - Construct an **algorithm** to solve the problem: a clear, step-by-step procedure that produces the correct output for any input.

Monday, 10/27/08, Slide #2
Printed 10/29/2008 1:33 PM

Problems

- Problem -- Has two parts:
 - Input: A clearly specified set of instances
 - Output: The question to answer or task to do with those inputs
- Example 1:
 - Input: A finite list of names
 - Output: The list sorted into alphabetical order
- Example 2:
 - Input: A set of numbers
 - Output: The average value of the numbers

Monday, 10/27/08, Slide #3
Printed 10/29/2008 1:33 PM

Algorithms

- ❑ **Algorithm:** A finite list of precise instructions that will take any of the possible input instances, and produce the desired output for that instance.
- ❑ **Computer program:** translation of an algorithm into a computer language

Monday, 10/27/08, Slide #4
Printed 10/29/2008 1:33 PM

Properties Algorithms Must Have

- ❑ **Precision and Determinism:**
 - A uniquely determined, precisely defined first step
 - For each step, a uniquely determined, precisely defined next step
- ❑ **Finiteness (Termination):**
 - For any input, the algorithm must terminate after a finite number of steps.
- ❑ **Generality:**
 - The algorithm can be used for any value from the set of possible inputs
- ❑ **Correctness:**
 - For any input, the algorithm must produce a correct solution to the problem by the time it terminates.

Monday, 10/27/08, Slide #5
Printed 10/29/2008 1:33 PM

Example from text, pp. 182-183

- ❑ **Problem:**
 - Find the maximum of 3 numbers, a , b , c
- ❑ **Algorithm:** Uses a 4th variable "*large*" to record the maximum value:
 - 1. $large = a$ (means *large* is assigned the value a)
 - 2. If $b > large$, then $large = b$
 - 3. If $c > large$, then $large = c$
- ❑ Check on some sample values ("trace the algorithm")
- ❑ Does algorithm satisfy the properties given on previous slide?

Monday, 10/27/08, Slide #6
Printed 10/29/2008 1:33 PM

Text's Pseudocode version of previous algorithm

Finding the Maximum of Three Numbers

This algorithm finds the largest of the numbers a , b , and c .

Input: a , b , c

Output: $large$ (the largest of a , b , and c)

```
1. max3(a, b, c) {
2.   large = a
3.   if (b > large) // if b is larger than large, update large
4.     large = b
5.   if (c > large) // if c is larger than large, update large
6.     large = c
7.   return large
8. }
```

Monday, 10/27/08, Slide #7
Printed 10/29/2008 1:33 PM

Pseudocode

- Shorthand way to write algorithms
- Uses language similar to programming

- **Title & Description**
 - Finding the Maximum of Three Numbers
 - This algorithm finds the largest of the numbers a , b , and c .

- **Input & Output:**
 - Input: a , b , c
 - Output: $large$ (the largest of a , b , and c)

- **Variable names:**
 - a , b , c , $large$

- **Assignment:**
 - $x = 3$, meaning "x is given the value 3"
 - Also popular: $x \leftarrow 3$, or $x := 3$
- **Comments:**
 - // if b is larger than $large$, update $large$
- **Functional notation**
 - $max3(a, b, c)$ {
...
...
return($large$)
}

Monday, 10/27/08, Slide #8
Printed 10/29/2008 1:33 PM

If-then-else constructs

- Text uses indentation to indicate action when if-clause is true

```
if (b > large)
  large = b
if (c > large)
  large = b
```

- Some use brackets to clarify:

```
if (b > large) {
  large = b
}
if (c > large) {
  large = b
}
```

- Can also have **else**-clauses or **else-if** clauses:

```
if (a > 0 and b > 0)
  print ("both > 0")
else if (a > 0)
  print ("only a > 0")
else if (b > 0)
  print ("only b > 0")
else
  print ("both ≤ 0")
```

- **Equal symbol:** $==$

```
if (a == b)
  print ("a is zero")
```

Monday, 10/27/08, Slide #9
Printed 10/29/2008 1:33 PM

Repetition constructs (loops)

- **While loop** – repeat an action while some condition holds
 - **For loop** – uses a counter variable to repeat an action a specified number of times
- ```
□ i = 1
 while (i ≤ 10) {
 print (i)
 i = i + 1
 }
```
- ```
□ for i = 1 to 10
  print (i)
```

Monday, 10/27/08, Slide #10
Printed 10/29/2008 1:33 PM

Example from text, p. 184

Finding the Maximum Value in a Sequence

This algorithm finds the largest of the numbers s_1, s_2, \dots, s_n .

Input: s, n

Output: *large* (the largest value in the sequence s)

```
max(s, n) {
  large = s1
  for i = 2 to n
    if (si > large)
      large = si
  return large
}
```

Monday, 10/27/08, Slide #11
Printed 10/29/2008 1:33 PM
