Time Logs

• You should be filling these out every time you work on one of the programming assignments.
• Be sure to turn them in regularly. Turn them in today if you haven’t turned them in already this week.

Software Lifecycle

• Problem Analysis and Specification
  – What needs to be done?
• Design
  – How should it be done
  – Creation of a solution (algorithm)
• Implementation
  – Turn the algorithm into a program
• Verification
  – Does the program do what it is supposed to do?
  – Does it not do what it is not supposed to do?
• Maintenance
  – Change what the program does
  – Includes both bug fixes and modifications

Problem Analysis and Specification

• What it does:
  – Clearly defines problem - what is/is not being solved
  – Refines imprecise problem to one that is solvable given existing constraints
  – Constitutes an agreement on what is to be done
  – May discover problems
    • Inconsistency, vagueness, impossibility
  – Leads the way to the solution
  – May contain desirable and optional items
Problem Analysis and Specification
(cont.)

• What it does (cont.):
  – Should be specific enough to be testable, so you know if/when the problem has been solved
  – Often done inadequately
• What it doesn’t do:
  – Specify how to solve the problem
• Important parts of a problem specification
  – A list of inputs
  – A list of constants
  – A list of outputs

Problem Analysis and Specification
(cont.)

• Examples:
  – Yes: Should calculate change in dime/penny land
  – No: Should be fast
  – Yes: Should run in less than 10 seconds
  – No: Should use quicksort
• For the programming assignments, I will provide a problem specification
  – This specification may be incomplete
  – It is your job to analyze and understand the specification and refine it as necessary

Design

• What it does:
  – Clearly specifies how the problem will be solved
  – Allows developers to determine what resources will be needed to solve the problem
  – Hopefully solves all problems that could arise in the development of the software component
  – Is used as a recipe for doing the actual coding
• What it doesn’t do:
  – A design is not code and does not contain any code.
    • May contain pseudocode
  – A design is not specific to any language, although it usually is specific to a type of language
Design (cont.)

A software design typically has 3 parts:
1) Identification of the data objects that are required to solve the problem
2) Identification of the operations that must be applied to the data objects in order to solve the problem
3) Construction of a detailed sequence of steps (an algorithm) that specifies how the operations can be applied to the data objects to solve the problem

Implementation

• Once the design is complete, coding can begin
  – Given a good design, this should be very straightforward
  – All hard problems should have been worked out in the design stage
  – New hard problems should send the project (temporarily) back to the design stage

Implementation (cont.)

• Good code should be
  – Correct
  – Readable and Understandable
  – Modifiable
  – Ideally: Reusable
Implementation (cont)

- Good programs should be:
  - Well structured
    - Break programs into meaningful parts
    - Stress for simplicity and clarity
  - Well documented (commented)
    - Good comments before each program and/or function
    - Good comments before each important part of a program/function
    - Use meaningful identifiers (function and variable names)
  - Aesthetically pleasing
    - Space things out and use blank lines between logical blocks
    - Use alignment and indentation to emphasize relationships

Verification

- Each program and subprogram should be tested against its requirements
  - To see that it does what it is supposed to do
  - To make sure that it does not do what it is not supposed to do
- Tests should include correct and incorrect inputs
  - Even nonsense inputs
- Regression tests
  - Make sure that new changes don’t break old functionality

Maintenance

- Bugs are found that need to be fixed
- Requirements change
- Components are reused
- Enhancements are made
- Generally accomplished by repeating the first four steps
- Most software development effort is maintenance
Example: Problem Specification/Analysis

**Problem**: Write a program that, given diameter of a circle, computes the area and circumference.

**Description**: Compute and output the area and circumference of a circle given the diameter.

**Input**: ?

**Outputs**: ?

**Constants**: ?

---

Example: Problem Specification/Analysis

**Problem**: Write a program that asks for the number of quarters, dimes, nickels, and pennies that you have. Then compute the total value of your change and print out the number of dollars and cents. The preferred output format is $X.XX$

**Description**: Calculate the total value of the change that you have. Print it out as a monetary value ($ and cents).

**Input**: ?

**Outputs**: ?

**Constants**: ?