CS107
Introduction to Computer Science
Lecture 1
Introduction

Csci 107

• This class is a broad introduction to CS. The goal is to find out what CS is about and find out about its applications and impact in other disciplines.
• Step-by-step introduction into the art of problem solving using computers
• It does not assume previous knowledge of programming or computers.
• It does assume that you will keep the pace, work on the labs in a timely manner, come to the help sessions, etc
• Intended for majors and non-majors

Administrativia

• Lab access
  – Searles 128:
    • Mon-Friday 8am-5pm (unless class in progress) and 6-10pm
    • Sat, Sun noon-10pm
  – Searles 117: 6-10pm, Sat-Sun 12-10pm

• Study group
  – Leader: Richard Hoang’05
  – Time: TBD
  – Location: Searles 128

Resources

• Class webpage
  http://www.bowdoin.edu/~ltoma/teaching/cs107/spring06/
• Office hours: M, T, W after class
• Grading policy
• Syllabus
• Lab assignments
• Readings

What is Computer Science?

• Computer Science is the study of computers (??)
  – This leaves aside the theoretical work in CS, which does not make use of real computers, but of formal models of computers
  – A lot of work in CS is done with pen and paper! Actually, the early work in CS took place before the development of the first computer
  – Computer Science is no more about computers than astronomy is about telescopes, biology is about microscopes, or chemistry is about test tubes.
• Computer Science is the study of how to write computer programs (programming) (??)
  – Programming is a big part of CS...but it is not the most important part.
• Computer Science is the study of the uses and applications of computers and software (??)
  – Learning to use software packages is no more a part of CS than driver’s education is part of automotive engineering.
  – CS is responsible for building and designing software.
What is an algorithm?

- **Algorithm**: well-defined procedure that allows an agent to solve a problem.

- Example algorithms
  - Cooking a dish
  - Making a peanut-butter jelly sandwich
  - Shampooing hair
  - Programming a VCR
  - Making a pie

Example

Is this an algorithm?

- Step 1: Wet hair
- Step 2: Lather
- Step 3: Rinse
- Step 4: Repeat

Would you manage to wash your hair with this algorithm? How about a robot? Why (not)?

Algorithms

An algorithm must:

1. Be well-ordered and unambiguous
2. Each operation must be effectively executable
3. Terminate.

Algorithm for Programming a VCR

- Step 1: If the clock and calendar are not correctly set, then go to page 9 of the instruction manual and follow the instructions before proceeding
- Step 2: Place a blank tape into the VCR tape slot
- Step 3: Repeat steps 4 through 7 for each program that you wish to record, up to a maximum of 10 shows
- Step 4: Enter the channel number that you wish to record, and press the button labeled **CHAN**
- Step 5: Enter the start time and press **TIME-START**
- Step 6: Enter the end time and press **END-TIME**
- Step 7: This completes the programming of one show. If you do not wish to program anything else press **END-PROG**
- Step 8: Press the button labeled **TIMER**. Your VCR is ready to record.

Types of Operations

- **Basic operations**
  - Wet hair
  - Rinse
  - Turn on VCR

- **Conditional operations**
  - If batter is too dry add water

- **Repeat/looping operations**
  - Repeat step 1 and 2 three times
  - Repeat steps 2,3,4,…10 until batter becomes soft.

Example

- Problem: Given two positive integers, compute their greatest common divisor

  - Euclid’s algorithm:
    - Step 1: Get two positive integer values from the user
    - Step 2: Assign M and N the value of the larger and smaller of the two input values, respectively
    - Step 3: Divide M by N, and call the remainder R
    - Step 4: If R is not 0, then assign M the value of N, assign the value of R, and return to step 2; otherwise, the greatest common divisor is the value currently assigned to N
Algorithm

- How to come up with an algorithm?
  - Problem solving

- How to represent an algorithm?
  - In English??
  - In a programming language??

Coming up with algorithms..

- How do people think????

- Puzzle:
  - Before A, B, C and D ran a race they made the following predictions:
    - A predicted that B would win
    - B predicted that D would be last
    - C predicted that A would be third
    - D predicted that A’s prediction would be correct.
  - Only one of these predictions was true, and this was the prediction made by the winner.
  - In what order did A, B, C, D finish the race?

Example

- Problem: Adding two n-digit numbers

  7597831 +
  1287525

  ------------
  8885356

  How would you write an algorithm to solve this problem?
  Assume the basic operation is adding one-digit numbers.

Examples of problems

Here are some problems that we’ll think of during this class

- **Searching**
  - Given a list of student names, and a target name, find out if the name is in the list or not
  - E.g.: search name on Bowdoin website; search a phone number in the phone book

- **Matching**
  - Given two lists of symbols, find out whether one occurs in the other
  - E.g.: ACATTGTACATTG and CAT

- **Movie search**
  - Given a list of movie names, and a keyword, find ou all movies that contain the keyword

Expressing algorithms

- Is natural language good?
  - For daily life, yes…but for CS is lacks structure and would be hard to follow
  - Too rich, ambiguous, depends on context

- How about a programming language?
  - Good, but not when we try to solve a problem..we want to think at an abstract level
  - It shifts the emphasis from how to solve the problem to tedious details of syntax and grammar.

Pseudocode

- Pseudocode = English but looks like programming

- Good compromise
  - Simple, readable, no rules, don’t worry about punctuation.
  - Lets you think at an abstract level about the problem.
  - Contains only instructions that have a well-defined structure and resemble programming languages