Computer Science Foundations
Puzzle-Solving Workshop and Seminar

Episode 1—September 30
Fall 2013

Paul Pham and Neal Nelson

September 30, 2013

Welcome to the Puzzle-Solving Workshop and Seminar for Computer Science Foundations. The purpose of this workshop is to help strengthen your mental muscles. You may feel weak at first, but don't worry. We'll start gently and gradually increase the weight of the problem, and you'll have super intellectual strength in no time.

What you use your newfound strength for is up to you, but we *can* tell you that it will make you a better programmer. Even if all you want to do is write video games or make a website to track baby polar bears, you'll have to solve mathematical problems like whether those asteroids are in range of your lasers or the distance between two GPS coordinates (latitute and longitude). This workshop will definitely help you in the Discrete Math thread.

Conversely, programming will make you better at math! (It's like a virtuous cycle). After all, one of the things computers do best is crunch numbers, and with a program, you can calculate mathematical results faster than working them out by hand.

So let's get started.

0.1 Instructions

1. Write your name and Evergreen login at the top.
2. Read the problem. If you already know how to solve it, or you can figure it out right away
   (a) write the answer quickly
   (b) turn it in to the instructors
   (c) get the optional bonus problems and follow those instructions.
3. If you have never seen the problem before or don't know how to do it, find a partner in the same boat, and work together.

4. Answer all the problems to the best of your ability. If you get stuck, raise your hand and someone will come tutor you.
5. Turn in your problems to the instructors before you leave.

1 Problem 1

One day young Carl Friedrich Gauss was bored in school. His teacher, wanting to keep his students busy doing a repetitive task, asked everyone to add up all the natural numbers from 1 to 100. Although they complained, all the students dutifully set about adding 1 to 2 to get 3, 3 plus 3 equals 6, 6 plus 4 equals 10, well, you get the idea. (This was before they had computers).

Everyone did this except Carl Friedrich. At first, he stared at the problem in confusion, like anyone else. This is the first step to solving any problem, and it is important not to skip it. The second step is to use a fancy symbol (he chooses the Greek letter ξ) to represent the answer that you want.

$$\xi = 1 + 2 + 3 + \ldots + 100 \tag{1}$$

The $\ldots$ in the equation above represents where you would fall asleep if you actually tried to list the numbers from 1 to 100.

Can you help Carl Friedrich solve his problem without adding one hundred numbers together, thereby teaching his teacher a lesson?

(a) What is the sum of the natural numbers from 1 to 100?

Trying using the strategy of enumerating simple examples above, or try drawing a picture of ξ number of blocks in a suggestive way.[1]

If you can solve this problem, you've solved the specific case of Gauss's problem for $n = 100$. What is the solution for general n?

(b) What is the sum of the natural numbers from 1 to n, as an expression in n?

2 Problem 2

Every morning, Ada must decide whether to bike to school or not (and take the bus instead). Because she is a very logical woman, she uses the following rules to help her decide.

1. By default, she wants to bike to get exercise and slow global warming.
2. If there is thunder and lightning, she will not bike.
3. Unless she has a class at 10:00am, then she will still bike through lightning and thunder.

[1]Not *that* kind of suggestive. In a way that *suggests* the right answer, silly.

Because she has heard that programming can make her life easier, she writes the following Python program to help her decide. All the variables in this program are *Boolean*, meaning they can only take on the values `True` and `False`.

```
bikeToSchool = True

if (thunderAndLightning and not classAt10OClock):
       bikeToSchool = False

print str(bikeToSchool)
```

(a) What is the value printed out by this program when the input values are `thunderAndLightning = False` and `classAt10OClock = True`?

(b) What is the value of `bikeToSchool` at the end of the program for all possible combinations of values `thunderAndLightning` and `classAt10OClock`? Draw a truth table.

(c) Write an equivalent program by changing the first three lines. This time, start with the default condition (first line) `bikeToSchool = False`.

Let's add another rule: if Ada is tired, she will not bike no matter what. We'll represent this with another Boolean variable, `isTired`.

(d) How would you modify the program above to make a decision for Ada's three rules?

(e) How many different combinations of values are there for all three Boolean variables?

(f) Let's *generalize* the problem. How many different combinations of truth values are there for n Boolean input variables?