High School Design Programming Competition:
Rules, Resources, and Examples

Engineering SELF Fellows
2016

1 Introduction
2 Rules
3 Concepts
   3.1 Assignment and Arithmetic
   3.2 Conditionals
   3.3 Loops
   3.4 Data Structures
   3.5 Functions and Methods
   3.6 Going Forward
1 Introduction

Welcome to the High School Design Programming Competition! This competition has been designed to test your knowledge of programming as a tool for general problem solving in your choice of the following languages:

- C++
- Java
- Python (v. 2.7 and 3.4)
- Ruby

If you have questions regarding the competition, please feel free to contact:

Ashley Hutton
Computer Science Undergraduate
Programming Competition Director
[ashley.hutton@ku.edu](mailto:ashley.hutton@ku.edu)
2 Rules

You will be given a docket describing a total of 7 programming challenges. Problems will be arranged in ascending difficulty and you will be allotted two and a half hours, in a single block of time, to write solution programs for these challenges.

An automated system called DOMJudge provides a website interface where you can submit your solutions for judging, receive feedback, and check a running scoreboard and clock. Before the actual day of the competition, a test competition will be available. Here, you can test submitting your programs for automated judging. More details will be sent sooner to the time of the competition. We will also be sending out instructional videos to assist you in ensuring you are familiar with correct methods of input and output.

All problems will be described uniformly: An introduction to the problem will set up the scenario and describe what challenge you must write a program to solve. It will then describe how the input structure which your program will accept, and prescribe how the program’s output must be formatted in order to be read correctly. An example of input and output will be provided for you to check your program against. Note: when your program is actually judged, different input data will be used, but the formatting will be the same. Pay very special attention to the section describing how to properly write your programs to accept input from DOMJudge.

Each correct problem is worth 1 point, so whichever team completes the most problems out of 7 will be designated the winner. In the case of ties, time (as marked in seconds, running from the beginning of the competition) will be used as a tiebreaker. So if more than 1 team completes all 7 problems, the first team to have done so is the winner. You may submit your solutions for judging as many times as needed.

Teams may consist of up to three students.

You will not be allowed use of the Internet except for access to the DOMJudge web interface, but any books on your programming language may be used during the competition.

On the day of the competition, your team will be allowed one computer. All of these machines run Windows 7. Software which students may use to write and test their programs will be uniform, and submission to the DOMJudge system for judging is done through an Internet browser.
3 Concepts

In this section, we list the core programming concepts you should be familiar with in your language(s) of choice. All the problems day-of-the competition will be solvable using just the concepts mentioned here. However, as the problems get more difficult, solutions will take less time to draft, code, and test if your team is well practiced and quite familiar with your chosen languages. Although we list here the basic concepts necessary to compete, we encourage you to read more in depth about features of your language, so that you may be more creative and efficient in writing your solutions. You may already be familiar with the basic stuff. If so, treat what follows as a refresher, then get to practice! In our little example snippets below, the code is Java.

3.1 Assignment and Arithmetic

All languages have some manner of giving names to data and doing basic operations on that data. You should be familiar with how your language declares variables and assigns values to them. You should also know how your language performs arithmetic operations, like addition and multiplication. In particular, you will want to be familiar with ‘Modular Arithmetic’, an operation that works by finding the remainder when you divide one integer number by another. Other useful math functions are Ceiling and Floor, which round decimals up or down to the nearest integer. You should also know about all the different ‘primitive’ data types your language has. Most languages can natively represent strings of characters, single characters, Boolean values (true and false), integers, and decimal numbers. But for some languages, 3 / 4 may not be the same as 3 / 4.0. Knowing little quirks about your language keeps you from running into odd errors.

3.2 Conditionals

Conditional branches allow you to evaluate logic – in particular, they allow your program to change the instructions it will execute depending on the kind of data your program was given. The most common way to represent conditional choice in a program is with if else statements. In general, when your program reaches a conditional block of code, it determines whether or not the condition following the if is true. For example:

```java
1 int x = 4;
2 if (x == 5) {
3     System.out.println("x is five!");
4 } else {
5     System.out.println("x is not five, bummer");
6 }
```
Here, we let our integer x be 4. When we reach our conditional, we check if x equals 5. If it does, we'd print out “x is five!”, but in our case, x does not equal 5, so instead we would evaluate the else portion and print “x is not five, bummer”. Most languages will also let you chain several if else statements together,

```java
String ourSentence = "Coi ro do!";
if (ourSentence.equals("Hi there!")) {
    System.out.println("Hi right back at ya!");
} else if (ourSentence.equals("I have to go...")) {
    System.out.println("See ya later!");
} else {
    System.out.println("Sorry, I don’t speak Lojban!");
}
```

Above, we use Java’s special equals method to check if a string is the same as another string. So there are lots of things that can be used to create a conditional statement in an if else block. And on top of that, you can put together several conditionals using logical operators || for or and && for and. For example, if (x * 2 == 4) || (x >= 10) reads as if x times 2 is equal to four or, if x is greater than or equal to 10. So for conditionals, you should know how your language writes if else blocks, what methods or operations can check if something is true or false, and how conditionals can be linked together using and and or.

3.3 Loops
Loops let you repeat a block of code again and again either a certain number of times, or until some condition (remember those?) is met. The basic loops for Java, C, and C++ are for, while, and do while. Python and Ruby have loops named the same, but they look and work slightly different. Review for your language what loops you have and how they work.

3.4 Data Structures
When we say Data Structures, we really just mean containers. Common containers in Java, C, and C++ include Arrays, Lists (or Vectors), and Hashmaps. Arrays and Lists are the simplest. Arrays are like a line of cubby holes, they have fixed length, and can be filled up with only one type of data at a time, like an Array of integers or an Array of Strings. Lists work similarly, but can grow in length. Both of these structures let you check what value was assigned to a particular index, and they let you assign values to indices. Hashmaps (also sometimes called Dictionaries) let you give pair a value to a keyword so that you can look up values if you know their keywords, or you can check if a value is in the hashmap. Arrays, Lists, and Hashmaps all have different ways of being used, and are useful at different times. You should be familiar with what Data Structures your language has and the ways they can be used. Arrays and Lists though will be your bread and butter for most problems at the competition. Note: In Python and Ruby, Lists are usually preferred to Arrays for the sake of simplicity.
3.5 Functions and Methods

All languages have special commands that either return specific pieces of information, or transform data in some way. For example, in Java, if we want to know the length of a string in the variable ourString, we could use `int stringLength = ourString.length();`, which would give an integer representing the length of the string. In Python, if we wanted to know the size of a list ourList, we could use `listLength = len(ourList)`. All the methods and functions of a language are described by the language’s Standard Library. The list of total methods and functions in the Standard Library is often quite large, but it’ll be fine if you just get familiar with those related to your language’s data structures and Strings. Most languages will also let you define your own methods and functions elsewhere inside your program. This can help your code keep clean if you find yourself repeating some pattern of operations multiple times, and knowing how your language declares and uses them may be helpful. In particular, you will want to be familiar with methods or functions in your language that work with strings and arrays (or lists for Python and Ruby teams).

3.6 Going Forward

To best learn and practice the concepts listed above, we recommend you check out the websites http://www.codecademy.com or http://www.codingbat.com. They’ll give you lots of simple challenges and tutorials that will introduce you to the languages supported by the competition. Each of them have lots of examples and even videos to help you learn the basic concepts in writing programs. Note: although these resources will help you learn aspects of your chosen language, we strongly encourage you to install an IDE (Integrated Development Environment) and grow familiar with how to write programs using it. Feel free to log into the test competition and submit test solutions for the problems included there.

For each language supported by the competition, there are several popular IDEs, and the computers provided to you to write and submit your solutions will have many available. The most common is Eclipse, which has great support for Java and C/C++. For Python, there’s Idle, and for Ruby, there’s Aptana. All of these IDEs are freely available. Naturally, you should use whichever you feel most comfortable with. If you would like to ensure your IDE or text-editor will be available to you, feel free to email the Programming Competition director, whose email is on the second page of this packet. We’ll see if your requested IDE is already installed, or if it can be.