UIL COMPUTER SCIENCE CONCEPTS

THE FIRST 15 – 2nd Edition

Written by

Kirby Rankin

Edited by Nancy Barnard

Author Kirby Rankin brings over 25 years of teaching experience and has coached Computer Science for most of those years. His successes include three individual champions and six 2A team champions, and these were in a row from 2008 through 2013. He had many, many more competitors qualify for region and state over his years.

We are a small company that listens! If you have any questions or if there is an area that you would like fully explored, let us hear from you. We hope you enjoy this product and stay in contact with us throughout your academic journey.
~ President Hexco Inc., Linda Tarrant

HEXCO ACADEMIC
www.hexco.com
P.O. Box 199 • Hunt, Texas 78024
Phone: 800.391.2891 • Fax: 830.367.3824
Email: hexco@hexco.com

UIL COMPUTER SCIENCE THE FIRST 15 – 2ND EDITION

Introduction

This guide is written for students preparing to take the Texas University Interscholastic League Computer Science written exam. The first fifteen questions of each UIL Computer Science written exam are purposely more basic (easier) than the remaining twenty-five questions. This guide is intended to show you how to correctly answer the first fifteen questions. UIL provides a list of topics to be covered by the written exam at http://www.uiltexas.org/files/academics/UILCS-JavaTopicList1617.pdf. Within that document can be found a list of topics used to create the first fifteen questions on the test. Those fifteen topics will serve as the chapters for this guide.

A goal of this guide is to help students that are just beginning their computer science studies prepare to be competitive at a basic level. Each correct answer on the Computer Science written exam is worth six points. Each incorrect answer deducts two points from the score. Skipped questions are not penalized. So, if you can answer the first fifteen questions correctly, you will earn a score of ninety (90) points. **A detailed study of past results shows that m*ost of the time a score of ninety will earn a medal (6th place or better) at the district level.*** This is true ***almost all the time*** in the first five classifications (1A – 5A) and almost half the time at the 6A level. Not only does 6th place earn a medal for the individual competitor, 6th place earns four points towards the team academic championship for your school.

UIL Computer Science is not just an individual competition. There is a team component as well. Four students compete on a team and one-half of the team score is comprised of the total of the top three scores on the written exam. The remaining half of the team score comes from the team members' performance in the hands-on (programming) portion of the contest. How to succeed at the programming portion of the contest will be left for another guide.

To further illustrate the importance of successfully answering the first fifteen questions consider that if three members of a team can each score 90 points by correctly answering the first 15 questions, that team's written score will be 270 points. That is 270 points WITHOUT any points from the programming portion of the contest. A team score of 270 has placed third or better at the district level in almost every district in 1A through 5A and a significant number of times in 6A.

The reason 3rd place is significant is that Computer Science and CX Debate are the only two UIL Academic contest for which team points and team medals are given for 3rd place. Points are given as follows: 1st place team 20 points, 2nd place team 16 points, 3rd place team 12 points. It is easy to see that simply mastering the first fifteen basic concepts on the test can produce great results for you, your Computer Science team, and your school.

New to this Second Edition printing is all but two of the chapters following this introduction contain five practice questions at the end of the chapter. These five questions were created and arranged such that the first two are similar to questions you might find on Invitationals A and B, the third question is a District level question, and the fourth and fifth questions in each set are similar to what might be found on the Region and State exams. Fifteen chapters contain five questions each for a total of seventy-five. Understanding the concepts covered by these seventy-five questions is essential to your success.

There is another useful and important portion of this guide that cannot be overlooked. Appendix B contains a checklist of concepts that ***must*** be memorized. If you cannot put a check next to each of the items in that list, then you are not yet completely prepared to score 90 on the first fifteen questions. Use the Memorization Checklist to guide you to those concepts that, whether we like it or not, must be memorized.

A few words about what this guide is NOT are appropriate at this time. This is a GUIDE, not a textbook. This guide is not a complete and exhaustive discussion of the topics presented. Reading and understanding this guide is NOT intended as a course in computer science. Using this guide along with enrollment in a good Computer Science 1 course will enhance your success without a doubt. The more that you seek out and use alternative resources that are available; the more successful you will become. A great list of resources is available at http://www.uiltexas.org/academics/computer-science/resources. This guide will not address the topics or complexity needed to answer the remaining 25 questions on each written exam. To go beyond district and be successful at region and state will most definitely require that you gain a much deeper understanding of ALL of the topics covered by the entire test.

Success in the UIL Computer Science competition requires a great deal of hard work and dedication, even to just get the first fifteen questions correct! To be successful you will have to practice A LOT! To be successful you and your teammates will have to memorize A LOT! But, if you will read and study this guide, memorize what you are asked to memorize, practice often, and make use of all the resources available to you, you can learn a lot of computer science and maybe win a few medals along the way.

Let's get started!

Kirby Rankin

Literal Math Expressions

Arithmetic Operators

There are 5 arithmetic operators in Java. Addition (+), subtraction (-), multiplication (*), division (/) and modulus (%). The order of these operators is the same as it is in mathematics.

1. Parenthesis
2. Multiplication, division and modulus from left to right
3. Addition and subtraction from left to right.

Addition, subtraction and multiplication behave just as you would expect. Division works differently for different types of numbers and modulus returns the remainder after doing division.

Here are a few expressions and their values:

8 + 4 – 3 = 9	Add the 8 and 4 first then subtract 3.
9 – 3 * 2 = 3	Multiply 3 times 2 and then subtract the product from 9.
(9 – 3) * 2 = 12	Subtract 3 from 9 and then multiply the difference times 2.

Division and Modulus

Before we look at expressions with division and modulus you need to learn how those two operators work in Java.

When one or both of the operands (numbers that are being divided) are decimal numbers, division works just as it does in mathematics.

5.0 / 3.0 = 1.6666666666666667
8 / 7.0 = 1.142857142857143
2.0 / 5 = 0.4

However, when both of the operands are whole numbers, the result will be the whole number quotient.

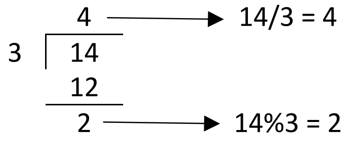
5 / 3 = 1 (3 goes into 5 one time with 2 left over)
15 / 4 = 3 (4 goes into 15 three times with 3 left over)

2 / 5 = 0 (💣**WATCH OUT!** This situation is the source of MANY missed questions on very easy problems. 5 won't divide into 2 and therefore the whole number quotient is 0.)

So what about the remainder? That's where modulus (%) comes in. The modulus (mod) operator returns the remainder resulting from division.

5 % 3 = 2
15 % 4 = 3
2 % 5 = 2

Here is an illustration using long hand division that shows the result of whole number division and modulus.



Practice Problems

11. What is the output of these lines of code?

```
out.println("The sky is blue ");
out.print("and there are\n clouds in the sky.");
```

A. The sky is blue and there are clouds in the sky.

B. The sky is blue
and there are clouds in the sky.

C. The sky is blue
and there are
  clouds in the sky.

D. The sky is blue
and there are\n
clouds in the sky.

E. The sky is blue
and there are
clouds in the sky.

12. What is the output of this line of code?

```
out.printf("The value of PI is %10.4f.",3.14159265359);
```

A. The value of PI is   3.1416.

B. The value of PI is 3.1416.

C. The value of PI is 3.14159265.

D. The value of PI is   3.1415.

E. The value of PI is 3.14159265359.

13. What is the output of this line of code?

```
out.printf("The distance to the moon is %,-5d miles.",238900);
```

A. The distance to the moon is -238,900 miles.

B. The distance to the moon is 238900 miles.

C. The distance to the moon is 238,9 miles.

D. The distance to the moon is 238,900 miles.

E. Error. Throws an IllegalFormatConversionException.

14. What is the output of this line of code? (# indicate blank spaces.)

```
out.printf("Total=%$12.2f",4321.9874);
```

A. Total=#####4321.99

B. Total=####$4321.99

C. Total=$####4321.99

D. Total=$4321.99

E. Error. Throws an UnknownFormatConversionException.

Code Blocks

The action to be taken if the controlling condition is true might require more than one line of code. If this is the case, all of the code that belongs to the if statement should be enclosed in ***curly braces*** to form a block.

In this next example, if the variable temp is greater than 100 we want to print a statement and we want to reassign a value of 0 to the variable and report the change. Take a look.

```
int temp = 118;
if(temp>100)
     {
     out.println("temp is greater than 100");
     temp = 0;
     out.println("temp has been reset to 0");
     }
out.println("current value of temp is " + temp);
```

will print

```
temp is greater than 100

temp has been reset to 0

current value of temp is 0
```

if we change the initial value of temp to 50 like this:

```
int temp = 50;
```

the output will become just

```
current value of temp is 50
```

💣 **WATCH OUT!** Forgetting the curly braces is a very common mistake and the UIL will test if you know what happens when you do forget them! Let’s look at what happens if we forget the curly braces in the previous example.

```
int temp = 150;
if(temp>100)
     out.println("temp is greater than 100");
     temp = 0;
     out.println("temp has been reset to 0");
out.println("current value of temp is " + temp);
```

prints

```
temp is greater than 100

temp has been reset to 0

current value of temp is 0
```

just as you would expect.