**Problets Terms**

**Problets.org** is the web site that serves problets.

A **problet** is short for “problem applet”. Each problet serves problems on one topic.

The **topics** covered by problets are the following (there is a separate problet on each topic for each of three languages: C++, Java and C#):

1. Arithmetic expressions
2. Relational expressions
3. Logical expressions
4. Assignment expressions
5. Bitwise expressions
6. If and if-else statements (called Selection)
7. Switch statements
8. While loops
9. For loops
10. Do-while loops
11. Advanced loop concepts including break, continue, infinite loops
12. Functions – debugging them
13. Functions – tracing their behavior (what does this function print when called?)
14. Arrays (1-dimensional)
15. Class/access concepts (public, protected, private)
16. Pointers in C++

You can try out the 46 problets at: http://problets.org/about/topics/index.html

Each topic is broken down into 9-25 **concepts**. In problets literature, concepts are also called **learning objectives.** The complete list of concepts is available at: http://problets.org/about/topics/learnobj.html Some concepts are:

* Precedence, associativity, correct evaluation of *each* operator
* Zero-iteration loop, independent and dependent nested loops (dependent means inside loop depends on the outside loop for its end condition), back-to-back loops for *each* type of loop (while, for, do-while)
* When debugging functions: non-void function without a return statement, value returned by non-void function ignored in caller, incorrect number/data type of parameters passed to a function, etc.

In a given problet/topic

On a given concept

5-25 **problem templates** have been encoded into the problet

For example, debugging functions problet might contain 10 problem templates dealing with “non-void function without a return statement”, and 8 templates dealing with “value returned by non-void function ignored in caller”. Each problem template has a unique **template number**, the concept for which it will be used, and the problem statement itself. The following table lists the number of concepts and total number of problem templates in each problet:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Topic** | **Used Since** | **Number of Problem Templates** | **Number of Concepts/Learning Objectives** |
| Expressions | Arithmetic | Fall 2004 | 192 | 25 |
|  | Relational | Fall 2004 | 268 | 24 |
|  | Logical | Fall 2006 | 280 | 21 |
|  | Assignment | Fall 2008 | 255 | 19 |
|  | Bitwise | Fall 2010 | 303 | 28 |
| Selection | If/if-else | Spring 2005 | 165 | 12 |
|  | switch | Spring 2010 | 147 | 12 |
| Loops | while | Fall 2004 | 201 | 9 |
|  | for | Fall 2004 | 213 | 10 |
|  | do-while | Fall 2010 | 125 | 15 |
|  | Advanced | Spring 2010 | 139 | 13 |
| Functions | Debugging | Fall 2009 | 117 | 9 |
|  | Tracing | Fall 2009 | 95 | 10 |
|  | Recursion | Spring 2013 | 68 | 10 |
| Arrays | 1-D | Fall 2010 | 172 | 14 |
| Classes | Access | Spring 2013 | 128 | 18 |
| **Total** |  |  | **2868** | **249** |

The **problem** presented to a student is an instantiation of a problem template. By instantiation, I mean, the meta-variables in the problem template are replaced with randomly chosen data types, names, literal constants, etc. All the problems generated from a problem-template are usually isomorphic, i.e., semantically equivalent.

In problets, there is a 1-1 correspondence between template number and concept. So, every student who solves a problem which is an instantiation of template 100 is dealing with the same concept. The exception to 1-1 correspondence are the five problets on expression evaluation: in those, a template may correspond to a set of concepts, e.g., 3 + 4 \* 5 corresponds to 3 concepts: the correct evaluation of addition and multiplication and precedence of multiplication over addition.

Every time a student uses a problet, the student goes through pre-test-practice-post-test protocol. So, *every* problet use is also an evaluation of problet.

* The sequence of problem templates used for pretest is fixed. It is the same sequence used by every student who uses that problet for a given programming language. As you might have guessed, there are some differences between the pretest of C++ and the pretest of Java/C#, because of differences between those languages, e.g., in C++, an assignment expression can be the condition of a loop, whereas Java will not allow it. So, C++ pre-test will contain a problem on this concept whereas Java pre-test will not.
* Practice is adaptive to each student. If a student solves all the pre-test problems correctly, the student is done with the problet – (s)he is not presented any practice or post-test problems. If a student solves pre-test problems correctly on 9 and incorrectly on only 4 of the 13 concepts, the student is presented practice problems on only those 4 concepts. Usually, the problems are presented two per concept in round-robin style, until the student has correctly solved a pre-set percentage of problems correctly on each concept.
* If the student demonstrates proficiency on a concept during practice, a post-test problem is scheduled for the student. If the student solves the post-test problem correctly, the student is done with that concept. On the other hand, if the student solves the problem incorrectly, additional practice problems on the concept are scheduled for the student.

Although every student initially solves the same sequence of pretest problems, the practice and post-test problems solved by the student and the order in which the student solves them vary from student to student. Since the use of each problet is limited to 30-40 minutes for logistical reasons, a student may also run out of time at any point in the protocol.

A **practiced concept** is a concept on which a student has solved problems during all three stages: pre-test, practice and post-test. A **learned concept** is one on which the student demonstrates pre-post improvement in score.

**DECA Analysis and Problets**

There is no concept inventory in Computer Science. One of the most recent articles on the topic

(http://www.tandfonline.com/doi/abs/10.1080/08993408.2014.970779) acknowledges this in its abstract. The concepts identified for each topic/problet are my educated guesses.

If it turns out that the number of dimensions of a pre-test is almost the same as the number of problems on the pre-test, empirically, the pre-test has minimal redundancy and the concepts covered by it are distinct/non-duplicative/minimally overlapping. That establishes an empirical basis for treating them as part of a concept inventory?

The above analysis can be compared against the analysis of practice problems, where multiple problems are presented on each concept on an as-needed basis. So, the number of dimensions should be far less than the number of problems.

Currently, problets assign all or nothing credit for each concept based on how the student solves the corresponding pre-test problem, e.g., if the student solves the problem 100% correct, the student gets full credit for the corresponding concept. 95% correct on the other hand, and no credit for the concept. If, for a given problem template in a given topic, we can repeatedly run DECA by variously partitioning the normalized grade space (0 – 1.0) and find the partition that yields the fewest dimensions among students who attempted that problem, then, I could use that partition point as the threshold for giving full credit for that problem template. This would benefit students in that they would end up solving fewer unnecessary problems. On the other hand, if no such partition can be found for a problem template, may be the problem is poorly framed?

Since students go through the same sequence of pre-test problems every semester on any given problet and programming language, DECA analysis should yield the same results for all semesters considered individually for a given problet/topic. If it does not, are students changing? Or does DECA analysis need to be re-worked?

Finally, the collected data of each problet in each semester can be partitioned by various criteria:

* men versus women
* undergraduates versus high school students
* CS majors versus non-CS majors
* traditionally represented versus underrepresented racial groups
* control versus test groups for a treatment
* C++ versus Java users

Would there be any difference in the DECA results for the two subgroups in each of the above comparisons? If so, what does it mean?