Math CountS
Pre-college Math Concepts versus Skills

Peter B. Henderson
Department of Computer Science & Software Engineering
Butler University
Indianapolis, Indiana 46208 USA
<phenders@butler.edu>

Some news to bring you up-to-date. I retired from Butler at the end of the Spring 2007 semester. So now I am a retired emeritus professor from two institutions: SUNY Stony Brook and Butler University. Accordingly, this has been an interesting summer as I ease into retirement. Before there was always pressure to get summer things done before starting the academic year. Now I find putting lots of them off knowing there is always tomorrow, next week, month, year, etc. The columns I write are also coming a bit more slowly.

I still plan remaining active in computing education and on preparing this column for the future. However, I hope others will step up to contribute articles since I will be missing important classroom experience where I can try and report on many ideas.

The contribution this month is from Maria and Gary Litvin, authors of the recently published book for high schools students *Mathematics for the Digital Age and Programming in Python*. Information about this unique text can be found at http://www.skylit.com/mathandpython.html. Here is an informative quote from the back cover *"The vision behind this book is that math and computer science should help each other. A programmer needs to be comfortable with abstractions, and that is precisely what math teaches. Computer science reciprocates by providing models and hands-on exercises that help clarify and illustrate more abstract math. Most importantly, both teach "precision thinking" — an important means of solving problems that call for exact solutions."* Maria and Gary are also the authors of *C++ for You++: An Introduction to Programming and Computer Science* (Skylight Publishing, 1998), which was a leading high school textbook for AP Computer Science courses, and the *Java Methods* series (Skylight Publishing, 2001-2006).

Pre-college Math Concepts vs Skills - Preparation for Computing Studies

Maria Litvin and Gary Litvin
Skylight Publishing - http://www.skylit.com

For quite a while now we have been pondering two seemingly unrelated questions: Why are American students behind their international peers in math, and what kind of math background does someone need to become a good programmer? Of course, these questions have been discussed for a long time in academia, in the media, and in backyard-barbecue conversations among parents of school-age kids. We believe that the two questions are actually related to each other: both have to do with the interplay of teaching more abstract concepts and more concrete skills in math and programming.

When one compares math education in this country to that of other countries, the eye-catching difference is how much American schools emphasize skills. In American culture, math is largely viewed as a set of skills to be mastered. These skills are concrete, and they are supposed to be relevant to everyday problems and to students' future careers. A popular Algebra I text, for example, names two goals in a section on multiplication properties of exponents: "Goal 1 — learn how to use multiplication properties of exponents to evaluate powers and simplify expressions; Goal 2 — learn how to use powers and the exponential change equation as models in real-life settings." The "real-life" example shows a photo of the Great Pyramid of Cheops and the formula for its volume, which includes the squared base of the pyramid (an exponent, for sure), and the volume (91,000,000 cubic feet). The chapter summary gives a "What did you learn" bulleted list which includes "skills," "strategies," and "exploring data." What's conspicuously missing is "concepts."

In virtually all other developed countries (European countries, Japan, Russia, China, etc.) school math is presented as a set of concepts and established mathematical facts. The facts are accompanied by proofs, which elucidate how the concepts relate to one another. Math is regarded as a unique opportunity to train students in special ways of thinking; the particular skills and methods are secondary.

Ironically, the American emphasis on skills — supported by often contrived, disingenuous examples — has resulted in a culture in which math is widely disliked and considered irrelevant to peoples' lives. Math teachers struggle to justify to their students how this or that fact is relevant. In other countries, meanwhile, the emphasis on concepts and proofs has ultimately resulted in stronger skills.

Mathematics teaches us to appreciate the beauty of a rigorous argument. In the long run, this is more valuable than learning a lesson on how to solve today's practical problems. The skills-first approach dampens students' interest.

It is true, of course, that mathematics does not exist in a vacuum — its abstractions are rooted in practical knowledge accumulated over centuries, and the teaching of mathematics draws on examples and analogies from the world around us. At least it should. And this brings us to our second question: What kind of math does a good programmer need to know?

Typical answers to the above question range from "none at all" to "the more the better." In a more pragmatic approach, one might think of such math topics as number systems, Boolean algebra, or mathematical induction. But what is more important, we think, is something we call "precision thinking." This includes exposure to more abstract mathematical concepts, understanding of rigorous proofs, and experience solving problems — not necessarily "real-life" problems, but *real* problems that go beyond correctly applying recently learned recipes. Both mathematics and computer programming nourish the ability to think with precision, and both require it.

A programmer needs to understand abstract concepts (data structures, algorithms, recursion, invariants) while also possessing very concrete skills (dealing with language syntax, software components and tools, debugging and testing, etc.). A good math background supplies the abstract thinking part. Any math is useful, as long as concepts and proofs are taught and real problems are solved. It is better, of course, if the math background includes more modern topics that are related to the world around us: number systems, Boolean algebra and digital circuits, graphs, combinatorics, probabilities, and so on. These topics can be taught on an introductory level no more complicated than middle school algebra, yet they are rarely taught in our schools.

One can envision a course in which math topics and programming projects are intertwined. In fact, one of us (Maria) is planning to teach such a course in the fall of 2007. The hope is to bring together kids interested in math and kids interested in computers and everyone else through an appreciation for precision thinking. The math topics should provide the necessary mathematical foundation to future programmers, while the hands-on programming projects will provide a welcome relief from more abstract mathematics.

Can changes in math and programming education take root in our schools? We believe they can, but that will take time, leadership, and resources. CS teachers with math backgrounds should lead the way.