IC101 : Programming Methodology I

IMPORTANT REMARKS ON LABORATORIES & TUTORIALS

(Any time, All the time)

Just DO IT! *Don't regret later.*

[illegible] 23/7/94

PRACTICE, PRACTICE, PRACTICE

IC101 is a course that introduces and teaches *good programming techniques* and *methodology*. Learning to program is an activity much like learning to play tennis. None of the world class tennis player has reached their top rankings by reading books about the theory and practice of tennis (even though these are also important) – they all got there through *constant, unrelentless practice, coaching, and trial and error.* Also, even with lots and lots of practice, most people *still* do not become world class players. However, *many of them* do become *good enough to enjoy* playing tennis.

Likewise, to be good at programming and to enjoy programming, you must practice it often, with constant coaching, and with a lot of trial and error. You learn by first working on simple programs, and then work your way up to more difficult ones and the IC101 laboratory sessions are designed to help you do exactly that.

Practice makes perfect and many experts have said that the best way to the top is *"Practice, practice, practice"*. In programming, it is also important to have lots of practice on programs of all kinds, from small ones to big ones, from simple ones to intricate ones. Practice makes perfect and constant and persistent practice allow you to learn from your mistakes. It is also important to learn good and proper programming techniques, systematic ways to program, good program design, and effective problem solving. The IC101 labs are designed to assist you in these areas.

Therefore, all ISCS freshmen must take advantage of the weekly two-hour lab sessions to gain lots of practice in programming. However, as all of you will soon discover, these mere two hours sessions are not sufficient practice. How many good tennis player you know keeps in form by playing only two hours a week? Therefore, it is very important that all of you have more practice with programming either here in DISCS, or at home, or anywhere else.

PRACTICE IT YOURSELF

In tennis, it is *unthinkable* to ask somebody else to "practice for you". If somebody else practice, then that somebody else improves, not you. The same thing applies to learning programming. You cannot learn programming by asking someone else to do it for you; you must do it yourself to learn programming. Watching good tennis players play can be beneficial, but one must *practice* what they have observed before they acquire the skill. The same for programming – reading good programs can be beneficial, but one must also practice what you have seen.

Good programming is a habit that must be acquired.

[illegible]
27/7/1994

BAD HABITS ARE WORST THAN IGNORANCE

Tennis coaches have often said that they'd rather take a student who have never played the game before than to take a player who have learned on his/her own and has picked up some bad habits. The reason is *obvious* – the coach has to help the students *"unlearn"* their bad habit before proceeding to teach them proper techniques. In addition, those who have mastered another similar game (like badminton or table tennis that requires a different use of the arm/wrist/body) find it very difficult to *"switch"* to tennis.

The same thing applies to programming – students who have picked up bad programming habits and techniques are much harder to teach than those who don't know programming at all. (Habits die hard.)[1] Also, those who pick up opposing views on good programming habits and techniques will find it difficult to "adjust".

Therefore, students who already know programming should take special note! Be aware of your programming habits.

LABORATORY SESSIONS

There will be weekly laboratory-cum-tutorial sessions to help you master programming and program design. The laboratory sheet for each week will be given to you ahead of time. You should read it and the appropriate parts of [Tucker-Lab] so that you will come to the laboratory sessions prepared for action.

The first few sessions will cover mostly technical matherials, discussions on the lab problem, and program design and coding. The later sessions will also include some discussions of tutorial problems. Each lab may also have a short, simple quiz.

In a typical week, the student must complete the lab exercise (solving an algorithmic problem) and ***turn in the solution within the time allocated.*** Students who are unable to complete them should turn in whatever they have. They should then try to complete it as soon as possible after that.

TUTORIAL QUESTIONS

There will be three types of tutorial questions in IC101, namely,

(R) *Routine problems.* The majority of the problems are routine problems that gives you lots of practice with all aspects of programming. They help you understand the materials covered in the course. You must do these problem ***yourself*** and if you encounter any difficulties (after many attempts), you must get help on them as soon as possible.

(D) *Discussion problems.* There will also be a small number of discussion problems. Usually, they will test your understanding of the material, your ability to interprete the results, or to apply the results, techniques, principles, etc.

(A) *Additional problems.* These are more challenging problems for students to explore. These usually involve some careful thought, and some may even involve further readings/research into the materials outside the course.
[Students are encouraged to try these problems and turn in their attempts, successful or otherwise. Dr. Leong gives out a "☕ award" to good attempts to these problems.]

[1] Many of these people become the so-called *hackers* – programmers who will somehow, by hook or by crook, get something done, but usually the solution is unsystematic, undocumented, and not pretty.