# MC28 General Information

**Max Hailperin, Karl Knight; Fall, 1995**

## Purpose

In MC28 we continue exploring the perspectives and methods of computer science, particularly abstraction. We'll first encounter the notion of computations with changing state by looking "under the hood" at computers. We'll then extend the notion of state into other areas, such as the use of object-oriented programming to model systems of objects with changing state. The remainder of the course focuses on the design and implementation of programming languages.

## Prerequisites

The prerequisite is MC27 (taken in Spring, 1991, or later).

## Classes and Labs

Since the class and lab meet the same hour, we aren't constrained to strictly staying with Monday, Wednesday, and Friday for the class and Tuesday and Thursday for the lab. Instead, the syllabus shows five dates that are classes rather than labs even though they are on Tuesdays or Thursdays, and six dates that are labs rather than classes even though they are on Mondays, Wednesdays, or Fridays.

Classes will be held in OHS 320, usually on Mondays, Wednesdays, and Fridays; Max Hailperin will be the class instructor. Classes will be used for discussion of the days reading and related topics, problem solving, and similar activities.

Labs normally will be held in OHS 326 on Tuesdays and Thursdays. As mentioned above, there will be some exceptional labs that are on Mondays, Wednesdays and Fridays; those may need to be in OHS 329 due to calculus use of OHS 326. (I'll let you know in advance where to go.) Karl Knight will be the lab instructor. Lab periods will be used for working on lab assignments.

## Office Hours

I will be available in my office (OHS 303) on Tuesdays and Thursdays from 9:00 to 9:50, on Mondays and Wednesdays from 3:30 to 4:20, and by appointment. You may send me electronic mail at max@gac.edu or call me at extension 7466.

## Textbook

Our primary text will be *Structure and Interpretation of Computer Programs* by Harold Abelson and Gerald Jay Sussman with Julie Sussman, MIT-Press and McGraw-Hill, 1985. However, we will also use a variety of handouts, and for the first two weeks we'll be using chapter 11 of the draft manuscript of *Concrete Abstractions: An Introduction to Computer Science*, by Max Hailperin, Barbara Kaiser, and Karl Knight. This draft chapter will be handed out free of charge at the first class. In the syllabus, readings listed as being from chapter 11 are from this draft chapter—the Abelson and Sussman text doesn't have a chapter 11.

## Honor

Students are encouraged to discuss the course, including issues raised by the assignments. However, the solutions to homework and laboratory assignments should be individual original work unless otherwise specified. If an assignment makes you realize you don't understand the material, ask a fellow student a question designed to improve your understanding, *not* one designed to get the assignment done. To do otherwise is to cheat yourself out of understanding, as well as to be intolerably dishonorable.

Any substantive contribution to your solution by another person or taken from a publication should be properly acknowledged in writing. Failure to do so is plagiarism and will necessitate disciplinary action.

Naturally, your use of college computer facilities should also be honorable. Please do not think of electronic breaking and entering as an intellectual challenge. Not only is it illegal and unethical, but it isn't even very challenging. Most security weaknesses are well known by now, and any two-bit hacker can easily re-discover the same old tricks over again. If you are looking for a *real* challenge, please see Karl or me instead—we can suggest problems without known solutions, and which are socially valuable as well. Similarly, don't electronically burglarize because some over-restrictive security protection is getting in the way of your doing something reasonable. Just see either of us, and we will reduce the protection or increase your authorization as appropriate.

## Late Assignments

All homework and lab assignments are due at the beginning of class on the day indicated. Late assignments will be penalized one letter grade for each non-holiday weekday or fraction thereof late. However, no late assignments will be accepted after graded assignments or solution sets are handed out.

There will be no penalty for lateness directly caused by illness. Simply write "late due to illness" on the assignment, sign your name, and hand it in. If you have a prolonged illness or an illness at the end of the term, such that catching up is impossible, or if you are ill for an exam, see me about special arrangements. Other circumstances will be evaluated on a case-by-case basis.

## Grade Changes

Please point out any arithmetic or clerical error we make in grading, and I will gladly fix it. You may also request reconsideration if you believe that we have been especially unjust.

## Exams

The two intra-term exams will be conducted during regular class periods and are indicated on the syllabus. The final exam will be as scheduled by the registrar.

## Grading

I will provide you with a letter grade on each lab assignment, homework assignment, and exam. As a guideline, the components will contribute in the following proportion to the final grade:

| | |
|---|---|
| lab assignments | 1/4 |
| homework | 1/4 |
| intra-term exam 1 | 1/8 |
| intra-term exam 2 | 1/8 |
| final exam | 1/4 |

However, I reserve the right to subjectively adjust your final grade. Please see me if you have any question how you stand. Class participation is not graded; however, it allows you to find and repair the gaps in your understanding before doing the homework or exam, and thus can dramatically improve your grade.

## Style Guidelines

All homework assignments and lab reports should be readily readable, and should not presuppose that I already know what you are trying to say. In particular:

- Use full English sentences where appropriate (namely almost everywhere, including in mathematical proofs or derivations).
- Word-process or type your homework if you can. In any case, make sure it is legible.

Programs should be easily read as well. The example programs in the textbook illustrate some stylistic issues. Some additional points to remember are the following:

- Begin each program file with a comment block explaining the purpose of the program and identifying the author and date.
- In general, each procedure should be preceded by a brief comment specifying what the procedure does (*not* how it does it). This may include assumptions it makes about its inputs and guarantees it makes about its results. Occasionally the name of the procedure and of its arguments will adequately substitute for this.
- Use comments within procedures sparingly, making sure that they explain the code rather than echoing it. Comments should explain why the procedure does what it does, or why it works, or what a low-level operation means in higher-level terms.

Be sure your assignments are always stapled together and that your name is always on them.

## Computer Logistics

Assignments will be done using the same facilities as MC27: Scheme/Schematik on the NeXT workstations in OHS 326 and OHS 329. Note that OHS 329 is exclusively for drop-in use, while OHS 326 may be used on a drop-in basis only when there is no scheduled lab section in it. You may also use the NeXT workstations in the first floor academic computing area, but we request you do so only when the 3rd floor MCS facility is closed.

It is also possible to do many of the MC28 assignments on the new SGI computers, using the SchematiX programming environment, which is very similar to Schematik on the NeXTs. (The underlying Scheme implementation is Gambit Scheme rather than MIT Scheme; this will mostly matter in that the debugging tools are different.) Some assignments, however, will only be possible on the NeXTs.

### Accessibility

Please contact me immediately if you have special physical circumstances, e.g. impaired vision, which may affect the accessibility of the laboratory or other course components. I will do my best to facilitate necessary arrangements or resources.

## Syllabus

| Date | Reading | Topic | Hw/Lab Due |
|---|---|---|---|
| 9/6 | | Introduction; SLIM | |
| 9/7 | | Lab 1: SLIM assembly language programming | |
| 9/8 | 11.1–11.2 | The SLIM architecture | |
| 9/11 | 11.3 | Iteration in assembly language | |
| 9/12 | | Lab 1 (continued) | |
| 9/13 | 11.4 | Recursion in assembly language | HW 1 |
| 9/14 | | Lab 1 (continued) | |
| 9/15 | | Recursion in assembly language (continued) | |
| 9/18 | 11.5 | Memory in Scheme: vectors | |
| 9/19 | | Lab 1 (concludes) | |
| 9/20 | 3–3.1.2 | Assignment and local state | Lab 1 |
| 9/21 | | Lab 2: Programming a robot | |
| 9/22 | 3.1.3 | Benefits of assignment | |
| 9/25 | 3.2–3.2.2 | The environment model | HW 2 |
| 9/26 | | Lab 2 (continued) | |
| 9/27 | 3.2.3 | Frames as local state | |
| 9/28 | | Lab 2 (concludes) | |
| 9/29 | | More on the environment model | |
| 10/2 | 3.2.4 | Internal definitions | HW 3, Lab 2 |
| 10/5 | | Lab 3: The environment inspector | |
| 10/6 | | Review/catch-up | |
| 10/9 | | Test 1 | |
| 10/10 | | Lab 3 (continued) | |
| 10/11 | 3.3–3.3.1 | Mutable pairs | |
| 10/12 | | Lab 3 (concludes) | |
| 10/13 | 3.3.2 | Queues | |

(continued)

## Syllabus (continued)

(A star after the date indicates that there will be class on what would normally be a lab day or lab on what would normally be a class day.)

| Date | Reading | Topic | Hw/Lab Due |
|---|---|---|---|
| 10/16 | 3.3.3 | Tables | Lab 3 |
| 10/17 | | Lab 4: Printing list structure | |
| 10/18 | | Tables (continued) | |
| 10/19 | | Lab 4 (continued) | |
| 10/24 | | Lab 4 (continued) | |
| 10/25 | 3.3.4–p.252 | Digital circuit simulator | HW 4 |
| 10/26 | | Lab 4 (concludes) | |
| 10/27 | | Digital circuit simulator (continued) | |
| 10/30 | 3.3.4 (rest) | The simulator agenda | Lab 4 |
| 10/31* | handout | Object-oriented programming | |
| 11/1 | handout | Object-oriented programming (continued) | |
| 11/2 | | Lab 5: Adventures in the land of Gack | |
| 11/3 | handout | Inheritance and delegation | HW 5 |
| 11/6* | | Lab 5 (continued) | |
| 11/7 | | Lab 5 (continued) | |
| 11/8 | | Review/catch-up | |
| 11/9 | | Lab 5 (concludes) | |
| 11/10 | | Test 2 | |
| 11/13 | 4–4.1.1 | Metalinguistic abstraction | Lab 5 |
| 11/14* | | More on the meta-circular evaluator | |
| 11/15* | | Lab 6: Making the meta-circular evaluator talkative | |
| 11/16* | 4.1.2–4.1.3 | Evaluator data structures | |
| 11/17* | | Lab 6 (continued) | |
| 11/20* | | Lab 6 (continued) | |
| 11/21 | | Lab 6 (concludes) | |
| 11/22 | handout | Using EBNF to define languages | Lab 6 |
| 11/27 | handout | A special-purpose language | |
| 11/28* | handout | The language and its implementation | |
| 11/29* | | Lab 7: Implementing a special-purpose language | |
| 11/30* | | More on the language's implementation | HW 6 |
| 12/1* | | Lab 7 (continued) | |
| 12/4 | handout | Data representation and garbage collection | |
| 12/5 | | Lab 7 (continued) | |
| 12/6 | | Garbage collection (continued) | |
| 12/7 | | Lab 7 (concludes) | |
| 12/8 | | Review/catch-up; evaluation | HW 7, Lab 7 |