1 Prerequisites and description

Prerequisites: C or better in CMSC 212 (or CMSC 214), and in CMSC 250
Credits: 3 credits

This course is a study of programming languages, including their syntax, semantics, and implementation. Methods for describing language syntax to be covered include regular expressions, finite–state automata, and grammars. The semantics of programming languages and their runtime organization are covered. Different models of languages will be discussed, including procedural (e.g., C, Pascal), functional (e.g., OCaml, Scheme), rule–based (e.g., Prolog), and object–oriented (e.g., Ruby, Java, Smalltalk, C++). Language features such as scoping and binding of variables, higher–order programming, typing and type polymorphism, pointers, object inheritance, and exceptions are explored.

2 Instructor contact information

Name: Larry Herman
Office: 1111 A. V. Williams
Phone: (301) 405–2762

Office hours will be provided in a separate handout shortly.

3 Teaching assistants and TA office hour information

<table>
<thead>
<tr>
<th>name</th>
<th>duties</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asad Sayeed</td>
<td>teaching, 0101 &amp; 0102</td>
<td>asayeed</td>
</tr>
<tr>
<td>Martin Paraskevov</td>
<td>grading</td>
<td>martin</td>
</tr>
<tr>
<td>Vasile Gaburici</td>
<td>grading</td>
<td>gaburici</td>
</tr>
<tr>
<td>John Dickerson</td>
<td>undergraduate– office hours</td>
<td>jdicker1</td>
</tr>
</tbody>
</table>

Note that all of the TAs’ email addresses are “@cs.umd.edu”.

The TAs’ office hours schedule will be provided in a separate handout shortly.

While the TAs will provide assistance with assignments during office hours, you are responsible for developing and debugging your own programs. You should therefore not rely on the instructional staff to getting your project to work. Lower–level CMSC courses provide extensive debugging and development help in office hours, but upper–level CMSC courses expect students to complete projects with minimal extra help. Therefore in CMSC 330 we will provide less debugging help than some students may be used to. If you come in with a question you should expect to be pointed in the right direction, but then it will be up to you to finish solving the problem on your own.

4 Resources, class webpage, and class web forum

There are no required or recommended texts for this course this semester. Some students may find reference texts for the languages taught to be necessary. Suggestions and other sources of information will be provided during the semester.

Various course materials will be made available on the class webpage, which is at the following URL: www.cs.umd.edu/class/spring2007/cmsc330/.

Accessing the webpage will require an ID and password to be provided in class.

Important material may be discussed in the class web forum, to be discussed further in the first project. Students are be responsible for reading this forum daily beginning when the first project is assigned; directions will be provided then.

Programming will be done on the OIT Grace Cluster, grace.umd.edu. Students will use their own Glue accounts to access the Grace cluster and do coursework, so students who don’t have a Glue account should request one online immediately at www.oit.umd.edu/new.
5 Exam and final dates

Midterm exams will be held during the lecture time. The following dates are when the midterm exams are expected to fall, so at this time you should plan on having exams on these dates. Note however that if it becomes necessary these dates could be adjusted depending upon lecture progress during the semester or other factors, therefore these dates will either be confirmed or adjusted as necessary, to be announced in class. The final exam date appearing below is fixed however, and will be rescheduled only for students having another final at exactly the same time, or for students with more than three final exams scheduled on the same day. (The only students whose finals are at the same time as this course’s final are those also taking BIOM 301, BMGT 350, or EDMS 451.) If either of these situations applies to you, you must inform the instructor within two weeks of the final exam time being announced in class for any allowances to be made. Also please let the instructor know immediately if you have a conflict with any of the tentative midterm dates, or any other important date as the semester progresses.

- Exam #1: Thursday, March 1
- Exam #2: Tuesday, April 24
- Final exam: Saturday, May 12, 4:00–6:00 p.m., location TBA

6 Attendance, homework, and general grading policies

Students are responsible for all material covered, and all announcements, deadlines, policies, etc., discussed in lecture and discussion section, regardless of whether they were in class or not. Students are likewise responsible for all announcements and information appearing in the class web forum, even if they did not read the forum. It’s understood that students may occasionally have to miss class for various reasons, but office hours are not intended as a replacement for class attendance. Consequently, only students who typically and regularly attend class will receive assistance during office hours.

Some practice homework exercises and solutions will be provided during the semester; these will not be collected or graded, but are extremely important for testing your knowledge of the material in preparation for exams. You are encouraged to work together on these problems. If you have difficulty solving them, see the instructional staff in office hours as soon as possible.

Coursework will count toward the final grade according to the following percentages:

- Programming projects: six expected coding assignments 40% (weighted as described below)
- Midterms: two midterms 30% (equally weighted; 15% each)
- Final: will be comprehensive 30%

Any request for reconsideration of the grading on any coursework must be submitted within one week of when it is returned. Exam regrading requests must be made in writing. Information about resolving any questions about project grading will be provided when the first project is graded. Any coursework submitted for reconsideration may be regraded in its entirety, which could result in a lower score if warranted.

Final course grades will be curved as necessary, based on each student’s total numeric score for all coursework at the end of the semester.

Note: this is a programming course, which teaches languages and concepts essential for later CMSC courses. As a result, the ability to submit working versions of the projects is necessary. Therefore, no student will be able to pass the course (with a grade of C– or higher) if at the end of the semester they have a zero grade for any project, regardless of their performance or scores on the other coursework. Complete project grading policies are below.

7 Project submission and grading policies

7.1 Project submission method and deadlines

Projects will be submitted electronically and directions will be provided with the first project assignment. No attempt to submit a project using any other means (such as an emailed project) be considered. Only the projects electronically submitted according to the procedures provided can be graded; it is each student’s responsibility to test their program and verify that it works properly before submitting.

All projects will be due at 10:00 p.m. on the day indicated on the project assignment. For every day (24-hour period) that a project is late a 10% late penalty will be deducted (i.e., 10% of the full project grade). Submission deadlines are firm and exceptions cannot be made. Note there is no grace period for project submissions—deadlines will be enforced at exactly 10:00:00 p.m. the day a project is due, and every 24 hours later.
Project extensions will not be given to individual students as a result of system problems, network problems, power outages, etc., so do not leave submitting a project until the night it is due. It is strongly suggested you finish and submit your program at least one day early, to allow time to reread the project assignment and all relevant articles in the class web forum, to insure you have not missed anything which could cause you to lose credit on the project.

7.2 Project grading policies

Projects will not be graded on style or documentation, although a certain design or structure, or certain language features, may be required, for some projects. During each project assignment certain minimum requirements which must be met will be specified. Projects will also be run on various other test cases which are not part of the minimum requirements and which will not be provided in advance; these will test conditions the minimum requirements do not.

Based on its results against the minimum requirements and other tests, and upon when it was submitted, a project will be graded as follows:

- A project which was not submitted will receive a score of zero (but see below).
- A submission which does not satisfy the minimum requirements for a project assignment, regardless of when it was submitted, will receive also a score of zero (but see below).
- A submission which satisfies the minimum requirements for that project will be graded out of 100 points as follows:
  - The 10% late penalty will be applied for each day late the project was submitted, up to a maximum 90 point deduction.
  - The project’s score will then be computed as follows:
    - correct results for the minimal requirements 60 points
    - results for other tests 40 points (prorated as described following)
    All credit for any input test cases which are not part of the minimal requirements will be lost if a project does not satisfy that input case; i.e., partial credit will not be given for individual test cases.
  - No project which satisfies the minimum requirements for that assignment will receive a score lower than 10 points (even if the late penalty and deductions for incorrect test cases would be greater than 90 points).
- If more than one submission was made by a student for any project (on-time, one day late, two days late, etc.) then all of them will be graded and returned, but only the highest score will be recorded.

As mentioned above, a student having a zero score for any project will not be permitted to pass the course with a grade of C– or better. However, project scores of zero may be removed, or replaced with a higher score, by correcting and resubmitting the project before the end of the semester. By the policies above, any project submitted nine or more days late which at least satisfies the minimum requirements for that project will receive a score of 10 points.

To summarize: a project score will be zero if no submission was made, or if one or more submissions were made but they all failed to satisfy the minimum requirements for that project. If a student has a score of zero for any project, the project must be corrected and resubmitted in order for the student to be able to pass the course with a grade of C– or higher. The sooner the corrected project is resubmitted (within eight days of the original due date) the more partial credit it can receive; if submitted nine or more days late a project which satisfies the minimum requirements will receive a score of 10 points.

Note that unlike lower-level programming classes, we will not be providing extensive automatic testing before projects are due and, other than satisfying the minimal requirements, projects will be graded based on test cases not provided in advance. Instead, you will be responsible for developing your own techniques for testing your own projects, and checking the correctness of your output yourself.

Lastly, any hardcoding in a project assignment will result in a score of zero for that project (which like any zero project score can be replaced by correcting and resubmitting the project, without hardcoding). Hardcoding refers to attempting to make a program appear as if it works correctly and actually calculates and computes correct results, when for some reason it actually does not do so. Examples would include a program which prints the desired output instead of computing it, or a program which works only because it takes advantage of properties which test cases in minimal requirements happen to have, etc. These are only a few examples; if you have any question about whether a particular situation would constitute hardcoding be sure to ask ahead of time.

7.3 Issues regarding computing resources

Projects can be developed on the OIT Grace UNIX Cluster. Either the Linux Grace hosts or the Solaris Grace hosts may be used. You may use any other available system, but all project submissions must work correctly using the language versions on the Grace cluster (details to be provided). Because different versions of languages may be installed elsewhere, a program may work perfectly on one system, yet not work at all on the Grace cluster. The program you
submit will be graded based on its results on the Grace Cluster, so having a working version on another system at any other time (or even another working version in your Glue account) can not be considered. No consideration in grading can be made for errors made in transferring files, or submitting the wrong version of your project. If you want to write any project on another system you are strongly recommended to complete it several days early, to have time to address any problems arising.

8 Excused absences and accommodations

8.1 Excused absences

Besides the policies in this syllabus, various University policies may apply to students during the semester. Various policies which may be relevant appear in the Undergraduate Catalog at www.umd.edu/catalog.

If you experience difficulty during the semester keeping up with the academic demands of your courses, you may consider contacting the Learning Assistance Service in 2201 Shoemaker Building at (301) 314–7693. Their educational counselors can help with time management issues, reading, note-taking, and exam preparation skills.

Missing an exam for reasons such as illness, religious observance, participation in required university activities, or family or personal emergency (such as a serious automobile accident or close relative’s funeral) will be considered to be an excused absence. However, students requesting an excused absence for any reason must apply in writing and must furnish documentary support for the assertion that the absence qualifies as an excused absence. For an absence due to medical reasons, for example, documentation would be from a health care professional who treated the student. In cases of illness simply being seen by a health professional is insufficient—medical documentation must state that a student was incapacitated and therefore unable to attend for an excused absence to be justified. For medical absences the documentation must include the phone number of the health care professional and must explicitly indicate the exact dates or times of incapacitation due to illness. The dates of incapacitation must include the date of the missed exam; an illness preceding an exam may not be considered to justify an excused absence. Self-documentation of illness is not sufficient support to excuse an absence. Excused absences will not be given unless documentation as described is provided. If you become ill, keep in mind that the University Health Center will not provide medical documentation.

It is the University’s policy to provide accommodations for students with religious observances conflicting with exams, but it is the student’s responsibility to inform the instructor in advance of intended religious observances. Written notice must be provided immediately upon an exam date being announced or confirmed in order for an absence to be excused, and if a known conflict exists with one of the planned midterm dates appearing below, notice must be given prior to the end of the schedule adjustment period.

When a student has an excused absence for an exam the score will be determined either by averaging their scores for the other exams (possibly a weighted average), or by giving a makeup exam. However, unless immediate notice is given as early as possible of the reason for any missed coursework, an excused absence may not be granted.

A student who might miss an exam for any reason other than those mentioned above must contact the instructor in advance to discuss the circumstances. An instructor is not under obligation to offer a substitute assignment or to give a student a makeup assessment unless the failure to perform was due to an excused absence.

The policies for excused absences do not apply to project assignments. Projects will be assigned with sufficient time to be completed by students who have a reasonable understanding of the necessary material and begin promptly. In cases of extremely serious documented illness of lengthy duration or other protracted, severe emergency situations, the instructor may consider extensions on project assignments, depending upon the specific circumstances.

8.2 Students with disabilities

Students with disabilities who have been certified by Disability Support Services as needing any type of special accommodations should see the instructor as soon as possible, during the schedule adjustment period.

All arrangements for exam accommodations as a result of disability must be made and arranged with the instructor at least three business days prior to the exam date, or accommodations will not be made.

9 Academic integrity statement

The Campus Senate has adopted a policy asking students to include the following statement on each examination or assignment in every course: “I pledge on my honor that I have not given or received any unauthorized assistance on this examination (or assignment).” Consequently, you will be requested to include this pledge on each exam and project.

Please carefully read the Office of Information Technology’s policy regarding acceptable use of computer accounts provided for instructional use at www.nethics.umd.edu/aug.
Note that programming projects are to be written INDIVIDUALLY, therefore cooperation or use of unauthorized materials on projects is a violation of the University’s Code of Academic Integrity. **Any evidence** of this, or of unacceptable use of computer accounts, use of unauthorized materials or cooperation on exams, or other possible violations of the Honor Code, **will be submitted** to the Student Honor Council, which could result in an XF for the course, suspension, or expulsion.

- For learning the course concepts (including the programming languages), students are welcome to study together or to receive help from anyone else. Students may discuss with others the project requirements, the features of the programming languages used, what was discussed in class and in the class web forum, and general syntax errors. Examples of allowable questions are “Does a Java class definition end in a semicolon?” or “What does a ‘class not found’ error indicate?”, because they convey no information about the contents of a student’s project solution.
- When it comes to actually writing a project assignment, other than help from the instructional staff a project must solely and entirely be a student’s own work. Working with another student or individual, or using anyone else’s work in ANY WAY except as noted in this paragraph, is a violation of the code of academic integrity and **will be reported** to the Honor Council. Students may not discuss design of any part of a project with **anyone** except the instructor or teaching assistants. Examples of questions which students may **not** ask others might be “How did you implement this part of the project?” or “Please look at my code and help me find my stupid syntax error!” Students may not use any disallowed source of information in creating either their project design or code. When writing projects students are free to use ideas or **short fragments** of code from **published** textbooks or **publicly available** information, if the specific source is cited in a comment in the relevant section of the program.

**VIOLATIONS OF THE CODE OF ACADEMIC INTEGRITY MAY INCLUDE, BUT ARE NOT LIMITED TO:**

1. Failing to do all or any of the work on a project by yourself, other than assistance from the instructional staff.
2. Using any ideas or any part of another person’s project, or copying any other individual’s work in any way.
3. Giving any parts or ideas from your project, including test data, to another student.
4. Allowing any other students access to your program on any computer system.
5. Transferring any part of a project to or from another student or individual by any means, electronic or otherwise.

If you have any question about a particular situation or source then consult with the instructor in advance. Should you have difficulty with a programming assignment you should **see the teaching assistants in office hours**, not solicit help from anyone else in violation of these rules.

**IT IS THE RESPONSIBILITY, UNDER THE HONOR POLICY, OF ANYONE WHO SUSPECTS AN INCIDENT OF ACADEMIC DISHONESTY HAS OCCURRED TO REPORT IT TO THEIR INSTRUCTOR, OR DIRECTLY TO THE HONOR COUNCIL.**

Every semester the department has discovered a number of students attempting to cheat on project assignments, in violation of academic integrity requirements. Students’ academic careers have been significantly affected by a decision to cheat. Think about whether you want to join them before contemplating cheating, or before helping a friend to cheat.

Students are welcome and encouraged to study and compare or discuss their implementations of the programming projects with any others after they are graded, **provided that** all of the students in question have received nonzero scores for that project assignment, and that the project in question will not be extended upon in a later project assignment.

### 10 Course topics (SUBJECT TO CHANGE)

The following list of lecture topics may vary according to the pace of lecture:

- Administrative and course introduction
- Scripting languages (Ruby)
  - Implicit vs. explicit declarations
  - Dynamic vs. static typing
  - Text processing and string manipulation
  - Data structures in Ruby
- Regular expressions and finite automata
- Environments, scoping, and binding
  - Functions and procedures
  - Parameter passing mechanisms
  - Dynamic vs. static scoping
  - Runtime implementations
• Functional programming (OCaml)
  – Lists and recursion
  – Higher-order programming
  – Types and polymorphism
  – Data types and pattern matching
  – Modules
  – Closures
• Parameters and scope
• Context-free grammars
• Programming language theory
  – Lambda calculus
  – Operational semantics
• Polymorphism and generics
• Concurrency
• Functional programming in object-oriented languages
• Pointers and garbage collection
• Exceptions
• Historical overview of programming languages

11 Right to change information

Although every effort has been made to be complete and accurate, unforeseen circumstances arising during the semester could require the adjustment of any material given here. Consequently, given due notice to students, the instructor reserves the right to change any information on this syllabus or in other course materials.

12 Copyright

All course materials are copyright Larry Herman © 2005. All rights reserved. Students are permitted to use course materials for their own personal use only. Course materials may not be distributed publicly or provided to others (excepting other students in the course), in any way or format.