Course Information

Welcome to CS 1101, your gateway to computer science. This course will introduce you to algorithmic and computational thinking, and programming in the small, and will address the following issues:

1. Problem formulation in a precise and concise fashion and independent of language considerations.
2. Design of simple algorithms from a problem specification, as well as correctness and analysis of efficiency.
3. The intermediate steps in the design of a program from an algorithm through a process of step-wise refinement. Language dependent considerations may be used in this process, but not elsewhere.

The emphasis throughout the course will be on the analysis required while designing correct and efficient algorithms. The course is intended to teach a student a systematic process of design - beginning with problem formulation from an informal specification, through convincing arguments to algorithms, the analysis of their correctness and efficiency, and finally arriving at programs through a process of step-wise refinement. A programming language bias will be avoided and programs will be developed in both imperative and functional styles. We will cover fundamental algorithmic ideas that have had (and continue to have) a major influence on the evolution of computer science and several other disciplines, ranging from mathematics to biology to the social sciences.

There will be three sections of this course. This handout describes basic course information and policies. Most of the sections will be useful throughout; the main items to pay attention to **NOW** are:

- ◇ Make sure you are signed up properly on AMS and on the course website.

1 Course Website

We will be using Gradescope to submit and grade assignments, and Piazza to post course material, answer questions, and for announcements, such as changes to office hours.

2 Staff

The lecturers for this course (3 sections) are Subhashis Banerjee and Debayan Gupta.

All contact with TAs and professors should occur via Piazza - ideally by writing to all instructors, not to any particular TA.

3 Prerequisites

This is an introductory course, so there are no prerequisite courses. We will expect familiarity with class 12 level mathematics (sets, relations, functions, basic logic and truth tables, basic counting and the principle of mathematical induction) – no prior programming experience is required, though it helps to be "computer literate".

This means that you should have taken mathematics in class 12; alternatively, you must clear the Calculus Enabler at Ashoka (with at least a 'B' grade).

4 Lectures and Coverage

Lectures will be held two days a week for 1.5 hours each. Check AMS to see the exact times for the section to which you have been assigned. You are responsible for material presented in lectures, including oral comments made by the lecturers. There may be make-up lectures and sessions held on specific topics from time to time. You are responsible for material presented in such sessions.

4.1 Coverage

Concept of an algorithm; recursion and principle of mathematical induction; correctness of algorithms; efficiency of algorithms - time and space measures; algorithms to programs; specification, top-down development and step-wise refinement; the notion of state and finite state machine; imperative programs; correctness and loop invariants; basic algorithm design techniques; encapsulation, abstractions and modularity; basics of object-

oriented programming; basic logic, soundness and completeness, example of a propositional resolution; an example of concurrency; an intro to numerical computation.

Note: The course will be programming language independent, and will involve programming in multiple programming languages, in both functional and imperative styles.

4.2 Lecture Outline

The first part of the course will introduce the basics of the functional and imperative models of computation, recursive and iterative processes, and the basics of programming using higher-order functions. Examples will be drawn from simple problems like factorial, computing x^n, computing sums and products, gcd, counting recurrences, fibonacci, simple memoization etc.

1. Introduction to the recursive model of computation (**2 hours**)
2. Recursion and correctness analysis using principles of mathematical induction (PMI) (**3 hours**)
3. Recursive programs in a functional language (preferably strongly typed, like SML, OCaml or Haskell) (**1 hour**)
4. Time and space complexity. Orders of growth. Big Oh. Simple recurrences. (**1.5 hours**)
5. Notion of a state. Iteration and invariants. Relation to PMI. (**3 hours**)
6. Imperative model of computation. State and memory. Iteration and the While-do loop. Invariants. Introduction to programming in an imperative language. (**1.5 hours**)
7. Procedural abstraction using higher-order functions (**3 hours**)

The second part of the course will introduce data-directed programming. It will emphasize on programming with records, lists, trees, arrays and developing abstract data types.

1. Notion of an *abstract data type*. Pairs (cartesian products, rationals). Signatures and classes. Objects. (**1.5 hours**)
2. Computing with lists: search, reverse, append, split, insert, insertion sort, recursive merge sort, quick sort. Continued emphasis on recursion, correctness and efficiency. (**3 hours**)

3. Computing with arrays: reverse, sequential and binary search, basic sorting: selection, insertion, bubble, merge, partition quick sort; other examples. Emphasis on correctness using array invariants and time complexity (**3 hours**)

4. Computing with trees. Simple examples (perhaps even one from graphs). Structural induction. (**3 hours**)

The third part of the course will address the issues in design and analysis of simple algorithms. Examples will be taken from Divide and conquer, Backtracking, Logical resolution, Numerical algorithms, Randomized algorithms and Geometric algorithms. Emphasis will throughout be on correctness and efficiency.

1. Depth first search and backtracking (**1.5 hours**)

2. Propositional logic and resolution using DFS. One or two examples in Prolog (**3 hours**)

3. Some examples of a complete applications (e.g. cryptography on a big number package) (**3 hours**)

4. Introduction to numerical computation. Floating point system. Polynomials and root finding (**3 hours**)

5. Basic idea of concurrency. Demonstration of a multithreaded program with locking (**1.5 hours**)

6. One or two slightly more advanced algorithmic and program development idea from parsing, syntax trees, graphs, geometry, divide and conquer, dynamic programming, . . .

5 Problem Sets

Problem sets will be assigned throughout the semester. Overall, we shall probably have 6 problem sets. The due date will always be written on the problem set itself. Homework must be turned in by 2 P.M. on the due date.

- **Late submissions:** We will not usually allow late submissions unless there are major extenuating circumstances, such as a medical issue.

  The only way to submit late is to email your solution to a TA and cc the instructor. We will sometimes allow small (read: 30 min) delays, but we intend to keep an eye out for habitual offenders and may reject such submissions out of hand.

- **Discount Policy:** Your lowest homework score will be ignored.
- **Office Hours:** There will be no office hours on the day a problem set is due.
- **Submission Format:** Solutions to written parts of the problem set should be submitted online to gradescope in a single PDF file. If the file does not clearly indicate which parts the solutions refer to, or has parts missing, it is assumed that the student did not attempt that part of the problem. Therefore, before submitting, make sure all of your work is included in the PDF file.

  Start each question on a new page and mark the top of the page with the following: (1) your name, (2) the question number, and (3) the names of any people you worked with on the problem (see Section 9), or "Collaborators: none" if you solved the problem entirely by yourself.

  The problem sets may include exercises that should be solved but not handed in. These questions will be clearly marked and are intended to help you master the course material. Material covered in exercises will be tested on exams.
- **Regrade Requests:** Any student who feels that a problem set was not graded properly may submit a regrade request through Gradescope within one week of the graded assignment being returned to the student. Please note the following before submitting a regrade request:

  1. You should carefully read the posted solutions for the problem in question.
  2. Indicate which rubric items you deserve (if applicable), where in your solution write-up you address them, and explain why you deserve extra points. Any regrades without justification will not be processed.
  3. The course staff reserves the right to regrade the entire assignment, and your grade may increase or decrease as a result of a regrade.
  4. **Important:** Lots of requests from the same person (hoping to somehow get extra points) and nonsensical requests will be dealt with harshly. This sort of thing wastes the time of the course staff and means that we can't help other students who might actually need it.

  If you are still unsatisfied with your grade after the regrade, please email Subhashis Banerjee and Debayan Gupta.

6 Exams

This course will have a midterm and a final exam:

Midterm: 1.5 hours (first week of March, just before spring break)

Final Exam: 1.5 hours (cumulative; during exam week)

The midterm and the final exam will be closed book. However, **you will be allowed to bring and use one double-sided, letter-sized piece of paper with your own notes for the first quiz, and two for the final**. These should not be necessary but might be helpful.

Attendance at the exams is mandatory. Legitimate conflicts can be discussed with the teaching staff but must be due to extenuating circumstances and discussed in advance. If a student misses either exam due to an emergency, make-up exams may be offered at the discretion of the instructor.

Invalid excuses: an interview; multiple exams on the same day; unprepared.

Regrade requests. Any student who feels that a quiz or final exam was not graded properly may submit a regrade request. The request must be made online by the announced deadline. The request should include a detailed explanation of why she or he believes that a regrade is warranted. Please make sure you read the solutions carefully before requesting a regrade.

7 Class Participation

The TAs will keep track of class participation by students; this will include both active participation during lecture and on Piazza.

8 Grading Policy

The final grade will be calculated as follows:

Homework	40%
Midterm	25%
Final exam	25%
Class participation	10%

9 Collaboration Policy

We encourage you to collaborate with your peers to deepen your understanding of the course material. However, you should approach collaboration on problem sets with care, and follow the guidelines below. Copying from online resources, books, or notes from

previous versions of this or other classes is strictly forbidden — such mindless behaviour will be considered a serious offense and dealt with accordingly. Using online resources *for information* is fine, as long as you cite them properly.

1. **You should spend at least 30–45 minutes trying to solve each problem entirely by yourself.** If you find yourself unable to solve the problem, you can seek help, either by approaching the TAs, or by using Piazza, or by discussing with your peers.
2. **Do not be a Spoiler.** If you already solved the problem, do not give away the answer to your friend. The best way you can help your friend is to give hints and allow her or him the pleasure of coming up with the answer her/himself. Our past experience has overwhelmingly shown that students who do not attempt the problem sets on their own generally perform poorly on the exams, and thus in the class overall.
3. **You must write up each problem solution entirely by yourself without assistance,** even if you discuss ideas with others to solve the problem. Doing otherwise will be considered plagiarism, an academic offense with serious repercussions.

   It is a serious violation of this policy to submit a problem solution that you cannot orally explain to a member of the course staff. Plagiarism and other dishonest behavior cannot be tolerated in any academic environment that prides itself on individual accomplishment.

If you have any questions about the collaboration policy, or if you feel that you may have violated the policy, please talk to one of the course staff. Although the course staff is obligated to deal with cheating appropriately, we are far more understanding and lenient if we find out from the transgressor himself or herself rather than from a third party or on our own.

Needless to say, **no collaboration whatsoever is permitted on quizzes or exams.**

10 Textbook

We will not use any textbook for this course.

11 On the Importance of Clarity

You should be as clear and precise as possible in your write-up of solutions. Understandability of your answer is as desirable as correctness, because communication of technical material is an important skill.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error, and because it is easier to read and understand. Sloppy answers will receive fewer points, even if they are correct, so make sure that your solutions are concise and well thought-out.

Sometimes, you will be asked to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of your essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudocode.
2. A proof (or indication) of the correctness of the algorithm.
3. An analysis of the asymptotic running time behavior of the algorithm.
4. Optionally, you may find it useful to include a worked example or diagram to show more precisely how your algorithm works.

Remember, your goal is to communicate. Graders will be instructed to take off points for convoluted and obtuse descriptions.

12 Advice and resources for effective learning

Because of the conceptual nature of the material, just attending lectures and doing the homework are unlikely to be sufficient for learning all the concepts. **Setting aside time to do the reading and to study your notes from lecture is generally necessary to truly learn and internalize the material**, and to be able to apply it in new ways later in the course as well as for the rest of your life.

Homework is essential for learning the material. Rather than thinking of problem sets as just a requirement, recognize them as an excellent means for learning the material, and for building upon it. Spread out the time you have to work the problems. Many people learn best by reading the problems long before they are due, and working on them over the course of a whole week; they find that their minds make progress working the problems in the background or during downtime throughout the day. Few people do their best learning the night before an assignment is due. Work with others if that is helpful, but with the goal of learning first and solving the problems second. **It is worth reading the posted homework solutions, even if you received full credit**. Often, the clarity of explanation or details of implementation are different from the way you were thinking about things in ways that can improve your learning.

Don't hesitate to ask for help. This class is largely conceptual, and the concepts tend to build on one another. **If you are having trouble understanding the material, it is important to catch up rather than risk falling further behind**. We can help.

Office hours are a particularly useful mechanism for learning material and working through difficulties on problem set assignments. Moreover, if you have questions about the course or problem sets, please use Piazza as opposed to emailing an individual TA or lecturer—that will give you a better chance of getting a speedy response.

This class has great material, so HAVE FUN!