Course title and number  CSCE 430 – Problem Solving Programming Strategies  
Term (e.g., Fall 200X)  Spring 2020  
Meeting times and location  Lecture MW 3:00pm-3:50pm HRBB 113  
Lab F 3:00pm-5:30pm ZACH 310  
Webpage  http://courses.cse.tamu.edu/keyser/csce430/  

Course Description and Prerequisites  
This course teaches methods for analyzing fundamental programming problems from a variety of domains and implementing solutions quickly and efficiently. The class will use problems based on competitive programming contests to develop skills in problem analysis, coding, and testing. Solving problems will involve identifying and applying a range of algorithmic solutions, including those dealing with combinatorics, dynamic programming, graphs, numerical calculations, string processing, and geometry, along with other more specialized algorithms.  

Prerequisites: CSCE 411 or Permission of Instructor  

Learning Outcomes or Course Objectives  
At the conclusion of this course, students should be able to:  
- Analyze a given programming problem to identify the algorithms needed to solve the problem  
- Implement a program, including implementing the basic algorithms needed, to solve specified problems  
- Develop test cases that will ensure that implemented programs are robust to a full range of valid inputs.  

Instructor Information  
Name  Dr. John Keyser  
Telephone number  979-458-0167  
Email address  keyser@cse.tamu.edu  
Office hours  TR 1:30-2:30  
Office location  HRBB 527C  

Teaching Assistants  
Name  Tanner Hoke  
Email address  hoke_t@tamu.edu  
Office hours  MW 4:00-5:30, EABB common TA area  

Name  Presley Graham  
Email address  presley.graham@tamu.edu  
Office hours  TBD, EABB common TA area
Textbook and/or Resource Material


The course will be using an automated online grading system, Kattis, that students will be required to sign up for. **There will be a $35/student fee for this system that you must pay directly yourself.** The details of how to pay and sign up will be given later.

Grading Policies

The course grades will consist of a large number of programming problems of varying difficulty and topics given over the course of the semester. Problems will be assigned in 3 different ways:

- Problems to be completed individually on the student’s own time. Generally, individual problems will be assigned each week and due on Saturdays, unless the class agrees on a different time. You should expect to spend several hours every week on your own, working on these problems.

- Problems to be completed within a timed period - lab. In each lab period, students will be given a set of problems to be solved during that lab period. These problems are meant to be challenging, with the time pressure being a significant factor.

- Team problems to be completed within a timed period - lab. During some of the lab portions of the class, students may be teamed up with one or two other students, and together they may be asked to complete problems during the lab period. In this case, all students on the team will receive credit for problems completed.

For each set of problems, students will be given a “base” number of problems they are expected to solve; there may be more problems given than that number, and students may solve more than the base number, if they are able to do so. The total number of base problems over the course of the semester will be approximately 125. In all cases, completing a problem will mean that the code passes a series of validated test data that the students will not be shown. Each problem will be noted as either complete or incomplete by the given time (either the due date or the end of the timed lab period).

Problems completed after the deadline (i.e. after the submission time for weekly individual problems, or outside of lab time during the timed period) can receive half credit. This is referred to as “upsolving.” Students will have a limited period to upsolve these problems late for half credit; typically this will be one week following the original deadline.

The final grade in the course will be based on the percentage of problems solved by each student (i.e. the number completed divided by the base number). That is, the number completed will be the number of problems completed individually on their own time, plus the number completed individually within a timed period, plus the number completed on a team within a timed period. The number possible will be the number of problems assigned as a base for students’ own time, plus the number assigned as a base during the individual timed periods, plus the number assigned as a base during team timed periods. All problems will be weighted equally.
To earn an A grade, a student must also get at least 50% of the base points for the problems to be completed individually on their own time, in every week of the class. This can include points for solving during the week, or later upsolving of the problems.

**Grading Scale**

The grading scale will be:

- **A** = 90 % or greater, and at least 50% of available problems in individual weeks
- **B** = 80-89 %
- **C** = 70-79 %
- **D** = 60-69 %
- **F** = <60 %

**Course Topics, Calendar of Activities, Major Assignment Dates**

**Schedule**

The following is the expected schedule, including some of the types of algorithms that are expected to be covered each week. However, as the semester goes on, this will likely be adjusted.

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction; Problem formats; Online judging systems; Parsing Input; Formatting Output</td>
<td>1</td>
</tr>
<tr>
<td>2-3</td>
<td>Fundamental Data Structures and their implementation on your own or in libraries (various trees, sets, graphs, search structures)</td>
<td>2</td>
</tr>
<tr>
<td>4-6</td>
<td>Applying Divide and Conquer, Greedy, and Dynamic Programming Approaches</td>
<td>3</td>
</tr>
<tr>
<td>7-9</td>
<td>Applying Graph Algorithms (search, shortest path, minimum spanning tree, network flow, bipartite graph matching)</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Applying Numerical algorithms and Combinatorics (GCD, LCM, Chinese Remainder Theorem and modular math, Large number computations, generating and counting permutations and combinations)</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>String Processing (editing, edit distance, subsequences, suffixes)</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>Applications of Geometric Algorithms (2D line segment and polygon queries – intersection, area; calculations on a sphere; 3D volume calculations; ray-surface intersection; convex hull; spatial subdivisions)</td>
<td>7</td>
</tr>
<tr>
<td>13-14</td>
<td>Selected additional algorithms and their application</td>
<td>8-9</td>
</tr>
</tbody>
</table>

**Other Pertinent Course Information**

**Reading**

Students may be required to read and implement algorithms based on the textbook material or other material provided online. Class time will include both presentation of new algorithms and concepts, as well as review of problem material from prior problem sets, but not all new material might be presented in class time.

**Online judging**

Online judging systems will be used to perform the testing and acceptance of solutions to problems. This will require students to obtain a user account on these external systems. The Kattis system (tamu.kattis.com) will be the main system used, but others might be used on occasion.
**Communication**
Communication will come through multiple routes beyond the classroom. Piazza will be the primary source for online discussion and announcements. A class webpage will be maintained with information about class schedule and problem set assignments. Ecampus will be used to post grades. You are responsible for checking all locations regularly.

**Absences**
University excused absences will be handled consistent with Student Rule 7 [http://student-rules.tamu.edu/rule07](http://student-rules.tamu.edu/rule07). Students missing a timed individual or team event for an excused reason will not have the base score for that period used in their grade calculation; any problems solved late for half credit will add to both the base and the solved portions of the grade. For the individual problems that students do on their own time, it is expected that students will work on these problems throughout the period of time they are assigned, so that any absences of less than 2 days should not affect the student’s ability to complete the problems on time. If the student has an excused absence of more than 2 days during the period for which the problems are assigned, the student will be given a number of days equal to the length of the excused absence, less two days, following the original due date or the student’s return (whichever is later). For example, a student who is sick for 4 days during a week would have the deadline for individual problems extended by 2 days. Note that the period for solving problems after the deadline for half credit might not be extended, since coded solutions might be posted at that time.

**Limitations on Anonymity**
The course will use a competitive programming framework for judging and posting results of all problem sets. This means that students will be able to see an indication of which problems other students have attempted, and their success or lack of success in having these accepted. While there can be the option of hiding the specific user name, students cannot expect their performance to be kept completely anonymous from others, and it may be that other individuals can determine their performance on problem sets either by process of elimination, or from other information they determine from the competitive system.

**Source Code Presentations**
Some problems worked on may have multiple solutions, and throughout the course, individuals’ work might be used as examples for illustrating approaches to solving a problem, writing code, etc. Students may be called on to describe their own solution to a problem or the approach they tried; advance warning will be given in such cases with students given the opportunity to opt-out of presenting. Alternately, students’ code may be used as an example shown to other students for how a given solution might be coded; code presented this way will be kept anonymous unless the student has first agreed to let it be presented.

**Computers**
Students will need to bring their own laptop to the Friday lab sessions. The laptop should be sufficient to enable to students to write and test their own computer programs.
**Americans with Disabilities Act (ADA)**

The Americans with Disabilities Act (ADA) is a federal anti-discrimination statute that provides comprehensive civil rights protection for persons with disabilities. Among other things, this legislation requires that all students with disabilities be guaranteed a learning environment that provides for reasonable accommodation of their disabilities. If you believe you have a disability requiring an accommodation, please contact Disability Services, currently located in the Disability Services building at the Student Services at White Creek complex on west campus or call 979-845-1637. For additional information, visit [http://disability.tamu.edu](http://disability.tamu.edu).

**Academic Integrity**

The Aggie Honor Code is: “An Aggie does not lie, cheat, or steal, or tolerate those who do.” Upon accepting admission to Texas A&M University, a student immediately assumes a commitment to uphold the Honor Code, to accept responsibility for learning, and to follow the philosophy and rules of the Honor System. Students will be required to state their commitment on examinations, research papers, and other academic work. Ignorance of the rules does not exclude any member of the TAMU community from the requirements or the processes of the Honor System. For additional information please visit: [http://aggiehonor.tamu.edu](http://aggiehonor.tamu.edu)

For this course, a significant amount of work will require solving problems for which a solution or test data might be available or posted online. Unless otherwise specified, students are not allowed to seek out or examine code/data for these problems on their own, prior to turning in their own solutions. Likewise, students should not read explanations of how to solve a specific problem prior to the original due date (i.e. the original deadline); note that reading a description of the solution is allowed for problems solved after the deadline for half credit. Accessing unallowed information will be considered a violation of the honor code, and students caught doing so will be referred to the honor council, regardless of whether the actual code is copied or not. Given the ease with which it may be possible to cheat in this way, any violations should expect to receive the maximum penalty from the honor council.