Social and Behavioral Sciences
Cover Letter
Re: Submission of CSCE 110, 111, and 121 for inclusion in the core curriculum.

To the University Core Curriculum Committee:

The Department of Computer Science and Engineering, with the support of the College of Engineering, is submitting a set of three courses, CSCE 110, 111, and 121, for consideration to be included in the Social and Behavioral Science track of the core curriculum.

In a relatively short period of time, computing has fundamentally changed the way that people live on virtually every level: the way individuals work, remember, and make decisions; how people interact with each other; and how larger organizations relate to individuals and to each other. Computing has impacted virtually every field of academia and is a critical part of advances in many fields even beyond the "STEM" disciplines. While we realize we have some bias, we believe that having a clear understanding of the fundamentals of computing will be critical for students to operate effectively in the changing societal landscape and become leaders of the 21st century. We believe the classes we are proposing will offer students a valuable option within the core curriculum that will give them the tools to engage in this increasingly computer-centric society.

In examining the core curriculum foundational component areas, we believe that the best match for our courses is the Social and Behavioral Sciences area. Though this might not intuitively seem like the location for these courses, we believe that in fact our courses directly address the goals and objectives of this foundational area, as we outline in the curriculum forms.

The three courses we are proposing all share great similarities in goals, so the curriculum form notes will sound very similar. However, the three classes are structured slightly differently, and we wish to give a brief overview of those differences, here. The target audience for each class is slightly different, and this difference is reflected in the details of the projects that students work on, the particular programming language used, and some of the overall discussion in the class. However, any of the three courses would be appropriate for general enrollment throughout the university. None of the classes has a course prerequisite, although CSCE 121 has a prerequisite of prior programming experience for reasons explained below.

CSCE 110 has been developed primarily as a course for those who want an introduction to the basic ideas of computing and programming, but are less likely to engage in significant software development. CSCE 111 is also an introduction to computing and programming, but the topics are presented somewhat more formally with the aim of providing a foundation for those who may expect to deal with significant software in the future. CSCE 121 is designed for students who have had prior programming (e.g. through a high school course, or via CSCE 110 or 111, or just independent study), and who therefore do not need (and might be bored by) some of the basic material covered in one of the other courses. It still offers an introduction to program design and concepts of computing, but at a higher level that prepares people to continue on to higher-level Computer Science classes.

All of these courses share a similar goal, of introducing students to how software works, is developed, and is used to solve problems and enhance the way we live and work. We note that our fourth "introductory" course (CSCE 206), while also a course for those with no prior programming, is focused on teaching just programming skills, syntax, and technique; it is therefore not as good of a match for the core curriculum, and we are not proposing it to be included.
In past offerings, the enrollment in each class has often been limited by seats in the rooms. Assuming they are approved for the core, we anticipate offering additional sections of each class to handle what we believe will be increased interest.

These classes all draw significant interest from students outside our major. In reviewing the enrollment from 2013-2014, we counted the number of students with a Computer Science or a Computer Engineering major in each section (note that some students may have been double majors, or may have later changed major). For CSCE 110, 2% (4 out of 241) of students were CS or CE majors. For CSCE 111, 25% (64 out of 253) of students were CS or CE majors. For CSCE 121, 56% (260 out of 463) of students were CS or CE majors. It is somewhat more difficult to understand the Fall '14 enrollment numbers since Engineering now lists all incoming students as ENGE majors, but including ENGE students gives similar results this Fall (4% of CSCE 110, 36% of CSCE 111, and 53% of CSCE 121).

Note that we are simultaneously submitting changes to the course descriptions for these three courses, and we are including those files for reference, also. The changes to descriptions should better represent the material in the course and highlight its relevance to the core curriculum objectives. The syllabi included there are very similar to those being used in these courses this Fall. The material in CSCE 121 has changed somewhat, as the result of a year-long review in the past academic year through our Undergraduate Curriculum Committee, but the submitted syllabus reflects the revised course objectives.

We will be glad to discuss any details of the courses and of our proposal, and we look forward to the chance to present these for consideration by the committee.

Sincerely,

[Signature]

John Keyser
Professor and Associate Department Head for Academics
Department of Computer Science and Engineering

Attachments:
CSCE 110 Departmental Request for a Change in Course (w/syllabus)
CSCE 110 Core Curriculum Cover Sheet
CSCE 110 Core Curriculum Form
CSCE 111 Departmental Request for a Change in Course (w/syllabus)
CSCE 111 Core Curriculum Cover Sheet
CSCE 111 Core Curriculum Form
CSCE 121 Departmental Request for a Change in Course (w/syllabus)
CSCE 121 Core Curriculum Cover Sheet
CSCE 121 Core Curriculum Form
Texas A&M University

Core Curriculum Cover Sheet

Initial Request for a course to be considered for the Fall 2015 Core Curriculum

1. This request is submitted by (department name): Computer Science and Engineering

2. Course prefix and number: CSCE 110

3. Texas Common Course Number: n/a

4. Complete course title: Programming I

5. Semester credit hours: 4

6. This request is for consideration in the following Foundational Component Area:

☐ Communication
☐ Mathematics
☐ Life and Physical Sciences
☐ Language, Philosophy and Culture
☐ Creative Arts
☐ American History
☐ Government/Political Science
☒ Social and Behavioral Sciences

7. This course should also be considered for International and Cultural Diversity (ICD) designation:

☐ Yes ☒ No

8. How frequently will the class be offered? Fall, Spring, some Summer

9. Number of class sections per semester: 4-6

10. Number of students per semester: 150-200

11. Historic annual enrollment for the last three years: 257 ('11-'12) 230 ('12-'13) 241 ('13-'14)

This completed form must be attached to a course syllabus that sufficiently and specifically details the appropriate core objectives through multiple lectures, outside activities, assignments, etc. **Representative from department submitting request should be in attendance when considered by the Core Curriculum Council.**

12. Submitted by:

\[Signature\]

Date: 8/26/14

Approvals:

\[Signature\]

Date: 8/26/14

13. Department Head

\[Signature\]

Date: 8/26/2014

15. College Dean/Designee

\[Signature\]

Date

For additional information regarding core curriculum, visit the Texas Higher Education Coordinating Board website at www.thecb.state.tx.us/corecurriculum2014

*See form instructions for submission/approval process.*
Texas A&M University

Core Curriculum

Initial Request for a Course Addition to the Fall 2014 Core Curriculum

Foundational Component Area: Social and Behavioral Sciences

In the box below, describe how this course meets the Foundational Component Area description for Social and Behavioral Sciences. Courses in this category focus on the application of empirical and scientific methods that contribute to the understanding of what makes us human. Courses involve the exploration of behavior and interactions among individuals, groups, institutions, and events, examining their impact on the individual, society, and culture.

The proposed course must contain all elements of the Foundational Component Area. How does the proposed course specifically address the Foundational Component Area definition above?

Computing has become pervasive throughout society. Humans touch computing on a daily basis and increasingly rely on computing technology in every aspect of their lives. Computers have revolutionized the ways that people access and remember information, make decisions, interact with each other, and organize society. For people to thrive in this increasingly computer-centric world, it is important that they understand the basic ways in which computers work, interface with individuals, and can be controlled. In this course, students learn about the ways that computer software is designed and organized, the abilities and limitations of computer programs, how programs reflect communication between people and machines, how errors are found and dealt with, and how systematic approaches can be used to analyze the behavior of computer programs. While different populations interact with computers in different ways and thus will have different requirements for how computers are used, this course will expose many of the fundamental aspects of computer technology critical to having a deep understanding of how computers work.

Core Objectives

Describe how the proposed course develops the required core objectives below by indicating how each learning objective will be addressed, what specific strategies will be used for each objective and how student learning of each objective will be evaluated.

The proposed course is required to contain each element of the Core Objective.

Critical Thinking (to include creative thinking, innovation, inquiry, and analysis, evaluation and synthesis of information):

Software forms the basis for many of the ways individuals, communities, and societies interact today. A major part of this course deals with helping students learn to design and develop such software. Software development is primarily a problem-solving activity, in which developers create new and innovative solutions to well-formed problems within the constraints of the computer technology and the particular programming language being used. The development process involves understanding the various requirements for the software, design of a solution, implementation, testing and analysis of that implementation, and evaluation of the software for solving the given problem. In assignments for the class (both coding assignments and non-coding ones), students are assessed as to how well their design and implementation is able to solve the given problem. This measures how well they are able to analyze the problem (analysis), create a new solution to the problem (creative thinking, innovation), and think of and use appropriate tests for the program, making modifications as needed (inquiry, evaluation, and synthesis of information).

Communication (to include effective development, interpretation and expression of ideas through written, oral and visual communication):

While software can help facilitate communication among people and organizations, the software development process itself involves communication in multiple ways. Code is itself a form of (primarily written) communication between a programmer and the computer. Students in this class are also taught about program
Texas A&M University
Core Curriculum
Initial Request for a Course Addition to the Fall 2014 Core Curriculum
documentation, and how the code they write is a way of communicating to future programmers; they are assessed on how well they have structured and commented code to clearly communicate intent. The course includes topics related to developing user interfaces, a form of visual communication. Students learn the process for putting together graphical user interfaces, and are assessed on how well they have done so. Oral communication is a critical part of the software development process. Laboratory work in the class includes students presenting and discussing the code they have developed with others (this is a weak form of the formal code review process used in professional practice).

Empirical and Quantitative Skills (to include the manipulation and analysis of numerical data or observable facts resulting in informed conclusions):

The ways that people use computers today have an either obvious or somewhat hidden reliance on manipulation and analysis of numerical data. Reflecting this, many of the examples and assignments used in the course have a numerical component to them, in which programs are developed that can process and work with numerical data to perform some task. Students are directly assessed on how well they can do this. Furthermore, the testing and evaluation stages of the software development process involve examining the results of output to evaluate how well the software is meeting its desired goals, and possible revisions to address errors discovered in the process (an example of using both numerical data and observable facts to draw informed conclusions).

Social Responsibility (to include intercultural competence, knowledge of civic responsibility, and the ability to engage effectively in regional, national, and global communities):

Computers help facilitate interaction at all levels of society, from small groups through the global community. The course gives students familiarity with the basic concepts critical to the software that provides this interaction, and thus enables students to engage more effectively with their communities. The course includes material on software security, how errors can occur and the impact they can have, and how software can be designed so that errors that may still occur will have minimal societal impact. Assessment includes tests in which students are evaluated on their understanding of these issues.

Please be aware that instructors should be prepared to submit samples/examples of student work as part of the future course recertification process.
1 Course Objectives

1.1 Catalog Description
Basic concepts in using computation to enhance problem solving abilities; understanding how people communicate with computers, and how computing affects society; computational thinking; representation of data; analysis of program behavior; methods for identifying and fixing errors in programs; understanding abilities and limitations of programs; development and execution of programs.

1.2 Course Description
This is an introductory course designed for any student interested in using computation to enhance their problem solving abilities. No prior experience in programming is necessary. Students will use their problem solving abilities to implement programs in Python.

1.2 Learning Objectives
The objectives of this course for students to:
1. Develop a basic understanding of programming and the Python programming language.
2. See the value of programming in a variety of different disciplines—especially as it relates to your other college courses.
3. Appreciate the value of experimentation.
4. Be comfortable with the fact that there is more than one right solution to a problem.
5. Have fun!

1.2 Learning Outcomes
By the end of the course, students should be able to:
1. Identify how to apply computational approaches to investigate problems in several domains.
2. Recognize and interpret basic programs written in the Python programming language.
3. Design and implement basic programs in the Python programming language.
4. Design, execute, and evaluate programming-based experiments. Evaluate their results in the context of the problem being solved.

2 Course Material and Announcements

2.1 Class Lectures
Most of the material that you will need for this class will be presented in the class lecture. Thus, it is extremely important that you attend every class.
2.2 Course Webpage and eCampus
The definitive source for course announcements, reading assignments, reference materials, and class handouts is the course web page and eCampus.

- Course site: http://faculty.cse.tamu.edu/tlw/Courses/fall14/csce110
- eCampus: http://ecampus.tamu.edu

Please consult the course webpage and eCampus regularly as they will both be updated throughout the semester.

2.3 Course Topics
The major topics that will be covered during the 15 week semester are as follows. Each topic will be explored with numerous in-class examples followed by practice with lab assignments.

- What is programming? Why is it important?
- Simple Input/Output (raw input, print)
- Variables and Expressions
- Collective Data Structures (strings, lists, tuples, sets, dictionaries)
- Decision-making Statements (if, if/else, if/elif/else)
- Repetition / Loops (while, for)
- Commenting, layout, and structure of code
- Randomness
- Functions
- Simulation
- File-processing (CSV _les)
- Binary numbers
- Visualization (matplotlib, turtle)
- Finding and removing bugs and errors

2.4 Email
Occasionally, I will send email to the class. So, please make sure you check your email regularly.

2.5 Textbook
There is no textbook for this course. However, there are many good references that are available online that may find helpful while studying the material.

- Think Python: How to Think Like a Computer Scientist by Allen B. Downey
  http://www.greenteapress.com/thinkpython/thinkpython.html
- Dive into Python by Mark Pilgrim
  http://www.diveintopython.net

2.6 Programming Environment
We will be using the following programming environments for this class. All software used in the class is free for academic use and works on the major computing platforms (Mac, Windows, and Linux). The labs have already been setup with the following software. You are encouraged to install the software on your personal computers so that you can study effectively outside of class.

- Enthought Canopy Express
  https://store.enthought.com/downloads
3 Grading

3.1 Course Components
Your grade will be based on three components.

- **Exams (50%)** – There will be two, 75 minute exams and one 2-hour comprehensive final exam. Each of the 75 minute exams are worth 15% each for a total of 30%. The comprehensive final is worth 20%.
  - The dates of the two, 75 minute exams are not scheduled yet. I like to see how the class is performing with respect to assignments, quizzes, and class participation before scheduling an exam. Exam dates will be announced at least 2 weeks before the exam to give everyone sufficient time to prepare.
  - The final exam will be Friday, December 12th from 12:30pm to 2:30pm in HRBB 124.

- **Lab assignments/homework (25%)** – Lab assignments will be assigned weekly and are designed to help students understand the course material, provide practical programming experience, and help improve problem-solving abilities. As the course material increases in difficulty, it is expected that lab assignments will have to be completed outside of lab time. Lab attendance is required and will be used as part of the lab grade.

  If you have an excused absence (as defined by student rule 7 [http://student-rules.tamu.edu/rule07](http://student-rules.tamu.edu/rule07)), then you will receive credit for attending lab that day.

  Labs begin on September 8th for MW labs and September 9th for TR labs.

- **Weekly quizzes (25%)** – There will be weekly quizzes every Thursday consisting of questions concerning material in the lecture and the lab assignments. The purpose of the quizzes is to help you stay caught up on the lecture material in the class as well as test your understanding of the lab assignments. You will be expected to write programs by hand on the weekly quizzes.

  The first quiz will be on Thursday, September 11th.

3.2 Grading Scale
- 90% – 100% A
- 80% – 89% B
- 70% – 79% C
- 60% – 69% D
- 59% and below F

3.3 Academic Misconduct
- **Exams and quizzes.** If you are caught cheating on in-class exams and quizzes, you will receive an F and be reported to the Aggie Honor Office.

- **Lab assignments/homework.** Each assignment will be worth 100 points. If you are caught cheating on the lab, you will get 200 points deducted from your total lab score at
the end of the semester. If you get caught cheating on a lab assignment for the second
time, you will receive an F or F* in the course. In both cases, you will be reported to the
Aggie Honor Office.

As noted above, the penalties are steep if you get caught cheating or performing any
type of academic misconduct. So, please do your own work! You are more than capable
of doing the work required to be successful in this course. Thus, there is no need to
copy the work of others. (See Section 6 for more details on collaboration versus
cheating.)

4 Class Participation and Success

4.1 Asking Questions
It is extremely important for you to be engaged in the course. Otherwise, you will fall
asleep and wonder what happened to your tuition dollars. So, I encourage you to ask
questions during lecture. For the first few weeks, when asking a question, please stand
up and also state your name so that I know who you are.

4.2 Studying for the course
This is not an easy course because you are essentially learning a new language; a new
way of communicating your thoughts to the computer.

To meet our learning objectives, I will keep you busy with programming projects,
quizzes, and exams. You must be totally engaged in the course. But, more importantly,
if you get lost early in the course, it is really difficult (if not impossible) to recover. It's like
your math courses. If you don't know how to add, it will be impossible to learn how to
multiply. Learning to program is similar. As a result, there are multiple ways that
feedback is built into the course to help move you along in the course without getting
lost. However, you are responsible for putting in the time to study and seeking help
when you don't understand the material.

If you put in the time, this course is designed for you to succeed. How much time should
you be studying? One rule of thumb states that you should study 2 hours per credit
hour. Given that this is a 4 credit course, that translates into 8 hours of study. However,
I'll round that up to at least 10 hours per week outside of class lectures and labs once
we start writing more interesting programs.

5 The Good, the Bad, and the Ugly

5.1 The Good
This is one of my favorite courses to teach. My philosophy (which is shared with many
of my colleagues) is that learning to program (and learning computational thinking) is an
essential skill given the ubiquity of computing. So, the first good related to this course is
that you have a professor that is excited about teaching the material. The second good
is that you will learn a skill that will be useful to you in life but more immediately in your
other classes. And, maybe a third good is that you can understand your geeky friends or bosses (if you have such people in your life) better.

For many of you, learning to program will be uncomfortable at first, but the reward at the end is that you will feel a huge sense of accomplishment and a skill that hopefully transfers to other areas of your life.

5.2 The Bad
This is a large class. So, I will not have a chance to meet or know many of you personally. As a result of such a large class (over 120 students), there is a strict policy in relation to assignments, exams, and quizzes. Without the following four rules, the course would be unmanageable (e.g., assignments, quizzes, and exams would not be graded in a timely fashion) and pure chaos.

1. No late assignments will be accepted.¹
2. There are no make-up quizzes or exams.¹
3. Once a grade has been posted in eCampus, you have one week to make a written request to regrade an assignment, exam, or quiz if an answer that is 100% correct has been graded wrong. A solution that is partially correct will not be regraded.
4. Your overall grade will be based on the scores that you receive on exams, quizzes, and assignments.

5.3 The Ugly
Please do not lobby to have any of the four rules in Section 5.2 changed to suit your personal situation (e.g., assignment extension because you didn’t finish the assignment on time, makeup for a quiz because you overslept). All such requests will be greeted with the answer 'No.' Instead, spend your time on studying the material and enjoying the class.

6 Collaboration and Cheating on Lab Assignments
Given that this is the first programming course for many of you, it may be unclear as to what is acceptable or not acceptable when writing programs. The provided examples are meant to be illustrative and not exhaustive for lab assignments. In addition to what is specified here regarding collaboration and cheating on lab assignments, all aspects of the course (exams, quizzes, assignments) are covered by the academic integrity policies of the university as discussed in Section 8.

6.1 Acceptable forms of collaboration
In this course, you are encouraged to discuss topics together. For example, you are encouraged to study for exams together, and discuss methods for solving lab assignments. The main point to remember is to verbally discuss the high-level concepts needed to do lab assignments. You should not be speaking Python (or any programming language for that matter) to each other. And, why would you want to

¹ An emergency situation or university-excused absence is an exception to the rule. Please inform me of such situations as soon as possible to schedule a make-up for a missed exam or quiz or a new deadline for a missed assignment.
anyway? Finally, when discussing a problem with others, the software tools you use for writing your programs should be closed on your computer.

6.2 Collaboration that morphs into cheating
Four guidelines will help you stay safe as a collaborator and not hover dangerously close to academic misconduct.

- First, other than the instructor, TA, and peer tutors, it is never okay to look at the written work of another person or show another person your written work until after all grading on an assignment is completed. Also, remember to keep your eyes on your own screen. The moment you start looking at another student's source code you have crossed the line into cheating.
- Second, while you are discussing an assignment with another student, you should be discussing your concepts in English and at a high-level (e.g., drawing pictures). You should not be reading your program to another person.
- Third, you are not allowed to ghostwrite any assignment. Obviously, you cannot use ghost-writers for your own assignments.
- Finally, everyone in the class is expected to take appropriate measures for protecting their work.

6.3 Deterrents to cheating
In this course, there are several methods to keep you focused on your own work.

- Weekly quizzes will assess your understanding of the lab assignments. If you do amazing on the lab assignments and poorly on the weekly quizzes, then a red flag is raised for both you and the instructor. In order for these two scores to correlate with each other, you must complete and understand the lab assignments. One without the other will affect your grade negatively.
- Plagiarism detection software will be used on all assignments to make sure students have not copied code from one another. Any program that you submit must be your own work.
- If the work you submit is the result of academic misconduct, you will receive a severe penalty (such as F or F* in the course) according to Section 3.3.
- Remember, you do not have to cheat to be successful in this course. You are all more than capable of doing well if you put in the time and are open and enthusiastic about learning the material in the course.

7 Digital Etiquette Policy
Cell phones are not allowed in class. I understand that your phones connect you to your friends and family, but the classroom is a place where you connect with the professor and your classmates. You will learn more if you concentrate on the course while you’re in the course.

Laptops and tablets are allowed in the class under the following condition: you use these devices to interact with the course material. You should not use a laptop during class to play a game, check your friends’ statuses on Facebook, IM, respond to email, watch videos on YouTube, etc. Such activities not only distract you (you will be unable to participate meaningfully in the class), but also anyone around or behind you. If you often seem distracted by what’s on your screen, I will ask you to put your laptop or tablet away.
8 Academic integrity
The university's policies on academic integrity apply to all components of this course (e.g., exams, lab assignments, weekly quizzes).

8.1 Aggie Code of Honor
“Aggies do not lie, cheat, or steal nor do they tolerate those who do.”
Everyone is expected to attend all classes, complete assignments on time, and participate fully in class discussions and group projects. Violations will be handled in accordance with the Texas A&M University Regulations governing academic integrity. Visit http://aggiehonor.tamu.edu for more information.

8.2 Plagiarism
As commonly defined, plagiarism consists of passing off as one’s own the ideas, words, writings, etc., which belong to another. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turn it in as your own, even if you should have permission of that person. Plagiarism is one of the worst academic sins, for the plagiarist destroys the trust among colleagues without research cannot safely communicated. If you have any questions regarding plagiarism, please consult the latest issue of the Texas A&M University Student Rules, under the section for Scholastic Dishonesty.

9 Americans with Disabilities Act
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, in Cain Hall, Room B118, or call 979-845-1637. For additional information visit http://disability.tamu.edu.

10 Modifications to the syllabus
While not anticipated, there may be revisions to syllabus that are required once the semester begins. If this happens, the syllabus will be updated and students notified of the revision promptly. If a revision is made to the syllabus, the intent is that the modification will be advantageous to the student.

11 Tentative Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Resources/Handouts</th>
<th>Lab Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Course introduction</td>
<td>• Syllabus</td>
<td>• No lab this week</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Puzzle: How to Follow Directions</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Getting started with</td>
<td>• Slides: Getting</td>
<td>• Lab 1: Getting</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Started Slides: Decision Making and Repetition</th>
<th>Started Lab2: Python Expressions and IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Decision-Making and Repetition, the Guessing Game,</td>
<td>Lab 3: Decision-Making and Repetition</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Lists, Tuples, and Functions</td>
<td>Slides: Lists, Tuples, and Functions</td>
<td>Lab 4: Guessing Game and Rock, Paper, Scissors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Guessing Game program required for Lab 4.</td>
</tr>
<tr>
<td>5</td>
<td>Functions</td>
<td>integer-english.py</td>
<td>Lab 5: Pig Solitaire</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exam 1: list of topics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exam 1: sample questions</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Review for Exam #1</td>
<td>Labs only on Monday (2/21) and Tuesday (2/22) for exam review.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Functions, Modules, Simulation, Returned Exam</td>
<td>Answer key for Exam #1</td>
<td>Lab 6: Pig</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In class programs available on elearning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Simulation/Sets</td>
<td>Lab 7: English to Pig Latin Translation</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Monty Hall Problem/Sets/Files</td>
<td>Lab 8: Simulation (this lab is only available on elearning)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Files</td>
<td>Lab 9: Manipulating Files</td>
<td>Question #1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
|   |   |   | Template: q1-template.py
|   |   |   | Sample Data Files: text-files.zip
|   |   |   |   |
| 11 | Files | • Exam 2: list of topics
|   |   | • Exam 2: sample questions
|   |   | • Lab 10: Text Processing
|   |   | • A few files for Lab 10.
|   |   |   |   |
| 12 | Exam #2 Dictionaries, List Comprehension |   |   |
|   |   |   |   |
| 13 | Insider's View of a Laptop/Compiler vs. Interpreter/Binary representation | • Insider's View of a Laptop |   |
|   |   |   |   |
| 14 | Caesar's Cipher, ROT13, Course Wrap-up | • Course summary
|   |   | • Final Exam: list of topics
|   |   | • Final Exam: sample questions
|   |   | • Lab 12: Inverted Dictionaries and Binary/Decimal Representations (this lab is only available on elearning)
|   |   |   |
|   | Final Exam, 3-5 pm, HRBB 124 |   |   |
Texas A&M University
Core Curriculum Cover Sheet
Initial Request for a course to be considered for the Fall 2015 Core Curriculum

1. This request is submitted by (department name): Computer Science and Engineering

2. Course prefix and number: CSCE 111
   Introduction to Computer Science
3. Texas Common Course Number: n/a

4. Complete course title: Concepts and Programming
5. Semester credit hours: 4

6. This request is for consideration in the following Foundational Component Area:
   □ Communication
   □ Mathematics
   □ Life and Physical Sciences
   □ Language, Philosophy and Culture
   □ Creative Arts
   □ American History
   □ Government/Political Science
   □ Social and Behavioral Sciences

7. This course should also be considered for International and Cultural Diversity (ICD) designation:
   □ Yes  □ No

8. How frequently will the class be offered? Fall, Spring, likely Summer

9. Number of class sections per semester: 4-6

10. Number of students per semester: 150-200

11. Historic annual enrollment for the last three years: 229 ('11-'12) 235 ('12-'13) 253 ('13-'14)

This completed form must be attached to a course syllabus that sufficiently and specifically details the appropriate core objectives through multiple lectures, outside activities, assignments, etc. Representative from department submitting request should be in attendance when considered by the Core Curriculum Council.

13. Submitted by:
    Signature
    Date 8/26/14

14. Department Head
    Signature
    Date 8/26/14

15. College Dean/Designee
    Signature
    Date 8/26/2014

For additional information regarding core curriculum, visit the Texas Higher Education Coordinating Board website at www.thecb.state.tx.us/corecurriculum2014

See form instructions for submission/approval process.
Texas A&M University

Core Curriculum

Initial Request for a Course Addition to the Fall 2014 Core Curriculum

Foundational Component Area: Social and Behavioral Sciences

In the box below, describe how this course meets the Foundational Component Area description for Social and Behavioral Sciences. Courses in this category focus on the application of empirical and scientific methods that contribute to the understanding of what makes us human. Courses involve the exploration of behavior and interactions among individuals, groups, institutions, and events, examining their impact on the individual, society, and culture.

The proposed course must contain all elements of the Foundational Component Area. How does the proposed course specifically address the Foundational Component Area definition above?

Computing has become pervasive throughout society. Humans touch computing on a daily basis and increasingly rely on computing technology in every aspect of their lives. Computers have revolutionized the ways that people access and remember information, make decisions, interact with each other, and organize society. For people to thrive in this increasingly computer-centric world, it is important that they understand the basic ways in which computers work, interface with individuals, and can be controlled. In this course, students learn about the ways that computer software is designed and organized, the abilities and limitations of computer programs, how programs reflect communication between people and machines, how errors are found and dealt with, and how systematic approaches can be used to analyze the behavior of computer programs. While different populations interact with computers in different ways and thus will have different requirements for how computers are used, this course will expose many of the fundamental aspects of computer technology critical to having a deep understanding of how computers work.

Core Objectives

Describe how the proposed course develops the required core objectives below by indicating how each learning objective will be addressed, what specific strategies will be used for each objective and how student learning of each objective will be evaluated.

The proposed course is required to contain each element of the Core Objective.

Critical Thinking (to include creative thinking, innovation, inquiry, and analysis, evaluation and synthesis of information):

Software forms the basis for many of the ways individuals, communities, and societies interact today. A major part of this course deals with helping students learn to design and develop such software. Software development is primarily a problem-solving activity, in which developers create new and innovative solutions to well-formed problems within the constraints of the computer technology and the particular programming language being used. The development process involves understanding the various requirements for the software, design of a solution, implementation, testing and analysis of that implementation, and evaluation of the software for solving the given problem. In assignments for the class (both coding assignments and non-coding ones), students are assessed as to how well their design and implementation is able to solve the given problem. This measures how well they are able to analyze the problem (analysis), create a new solution to the problem (creative thinking, innovation), and think of and use appropriate tests for the program, making modifications as needed (inquiry, evaluation, and synthesis of information).

Communication (to include effective development, interpretation and expression of ideas through written, oral and visual communication):

While software can help facilitate communication among people and organizations, the software development process itself involves communication in multiple ways. Code is itself a form of (primarily written) communication between a programmer and the computer. Students in this class are also taught about program
Texas A&M University
Core Curriculum

Initial Request for a Course Addition to the Fall 2014 Core Curriculum
documentation, and how the code they write is a way of communicating to future programmers; they are assessed on how well they have structured and commented code to clearly communicate intent. The course includes topics related to developing user interfaces, a form of visual communication. Students learn the process for putting together graphical user interfaces, and are assessed on how well they have done so. Oral communication is a critical part of the software development process. Laboratory work in the class includes students presenting and discussing the code they have developed with others (this is a weak form of the formal code review process used in professional practice).

Empirical and Quantitative Skills (to include the manipulation and analysis of numerical data or observable facts resulting in informed conclusions):

The ways that people use computers today have an either obvious or somewhat hidden reliance on manipulation and analysis of numerical data. Reflecting this, many of the examples and assignments used in the course have a numerical component to them, in which programs are developed that can process and work with numerical data to perform some task. Students are directly assessed on how well they can do this. Furthermore, the testing and evaluation stages of the software development process involve examining the results of output to evaluate how well the software is meeting its desired goals, and possible revisions to address errors discovered in the process (an example of using both numerical data and observable facts to draw informed conclusions).

Social Responsibility (to include intercultural competence, knowledge of civic responsibility, and the ability to engage effectively in regional, national, and global communities):

Computers help facilitate interaction at all levels of society, from small groups through the global community. The course gives students familiarity with the basic concepts critical to the software that provides this interaction, and thus enables students to engage more effectively with their communities. The course includes material on software security, how errors can occur and the impact they can have, and how software can be designed so that errors that may still occur will have minimal societal impact. Assessment includes tests in which students are evaluated on their understanding of these issues.

Please be aware that instructors should be prepared to submit samples/examples of student work as part of the future course recertification process.
Computer Science 111  
Fall 2015

Instructor:
Dr. Bruce Gooch
Email: Gooch@cse.tamu.edu
Office: 438 Liberal Arts and Humanities Building (LAAH)
Phone Number: 801 828 5037
Office Hours: Monday Wednesday and Friday starting at 3:50 pm in the Harvey R. "Bum" Bright Building (HRBB) Room 113, we will then move to 438 LAAH and continue as long as needed.

Lecture Schedule:
Lectures are held Monday Wednesday and Friday from 3:00 pm to 3:50 pm in HRBB Room 113.

Laboratory Schedule:
Laboratory times are dependent on the class section a student is registered for.
Section 500 meets Mondays and Wednesdays from 4:10 pm to 5:00 pm in the Reed McDonald building (RDMC) room 111A.
Section 504 meets Tuesdays and Thursdays from 11:10 am-12:00 pm in RDMC room 111A.
Section 505 meets Mondays and Wednesdays from 1:50 pm to 2:40 pm in RDMC room 111C.

Catalog Description:
Computation to enhance problem solving abilities; understanding how people communicate with computers, and how computing affects society; computational thinking; software design principles, including algorithm design, data representation, abstraction, modularity, structured and object oriented programming, documentation, testing, portability, and maintenance; understanding programs’ abilities and limitations; development and execution of programs.

Course Overview:
The computer’s role in culture has expanded from a calculating machine used by governments to the iPod as a fashion accessory, the Smart Phone as a companion, and the Internet as a medium of self-expression. In the 1950s, the idea of dedicating a computer to entertainment was unthinkable; today revenues from the computer game industry exceed Hollywood. Today more humans own a computer than own a toothbrush and Apple is the world’s largest company having overtaken Oil, Agriculture and Manufacturing.

We inhabit a century where every job will be technical. In the 21st century, learning to program a computer is empowerment. Programming is the power to create, the power to change and the power to influence. Today’s students regardless of their ultimate field of study or occupation need this fundamental knowledge.

Programming teaches logic, algorithmic thinking, and an iterative approach to solving problems and testing your ideas. These skills make you smarter and are useful no matter what job you do. Learning to program can be incredibly rewarding, give you a new appreciation for technology and introduce countless career opportunities. There are two critical steps to become literate in programming:

1. Learn the Basics of Programming:
From “if” conditionals to “for” loops, knowing the basics of programming allows you to understand the way the modern world works. Programming instruction teaches procedural and functional thinking, project management and time management, skills that are essential components of an empowered individual.
2. Learn to Program for the Web:
Every one and everything has a presence on the web these days. Knowing the basics of HTML, CSS JavaScript and PHP as well as how they are used to create web pages are skills that will always be useful. Just like knowing a foreign language, these skills are helpful in all professions. They are also a must-have in many "non-technical" fields.

“I think everybody in this country should learn how to program a computer because it teaches you how to think.” The Lost Interview with Steve Jobs.

Topics:
In this course, we will study the technology, science, and art of programming a computer. Programming is the hands-on, inquiry-based way students will learn computational thinking and the principles of computer science. We will investigate the abilities and limitations of computer programs. We will learn how people communicate with computers, and how computing affects society. We will learn fundamental concepts for using computation to enhance our problem solving abilities including algorithm design, data representation, abstraction, modularity, structured and object oriented programming, documentation, testing, portability, and maintenance. We will also learn about software design principles used to develop computer code.

Course Objectives and Learning Outcomes:
Students completing this course should be able to:
• Use a text editor to program Java code.
• Use a Java compiler to produce executable software.
• Identify the basic syntax and semantics of the Java programming language.
• Use appropriate commenting, layout, and naming to communicate code intent.
• Recognize the primitive data types built into the Java language.
• Apply the concepts of Java classes and objects.
• Differentiate the difference between variables of primitive types and variables of class types.
• Contrast the concepts of lifetime, scope and the initialization mechanism of variables.
• Implement Java code branching using if statements.
• Program loops with while, for and do statements.
• Identify and correct syntax and logic errors in short Java programs.
• Apply the software engineering principles of encapsulation and data hiding by assembling data and methods into classes.
• Apply the principles of modularity and abstraction when organizing Java code.
• Create Java I/O interfaces, including graphical user interfaces, in a console or web-based system.
• Make use of arrays to store and process lists of data.
• Read, interpret, analyze, and explain introductory Java programs.
• Test and evaluate introductory Java programs.

Textbook:
“Learn Java the Hard Way” by Graham Mitchell
https://learnjavathehardway.org/

Students can buy a download code for “Learn Java the Hard Way” + the tutorial & answer videos at the campus Barnes & Noble. The cost will be $40 (a savings of $5).

Additional programming exercises can be found at Graham Mitchell’s website “Programming by Doing.” https://programmingbydoing.com/

Assignments
In this course there will be 60 assignments each worth 1% of the final grade. Students will be responsible for completing and turning in the 49 exercises in “Learn Java the Hard Way” and an additional 11
exercises from “Programming by Doing.” The assignments are an integral part of the course and you cannot learn the material without doing them. Many exam questions will be directly related to the assignments. Due dates are specified for each assignment as they are given throughout the course.

**Quizzes**
There will be 25 Quizzes given during the semester. Each quiz will be worth 1.5% of the final grade. Students will be allowed to drop their lowest five scores.

**Grading**

<table>
<thead>
<tr>
<th>Coursework</th>
<th>Weight (out of 100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>60%</td>
</tr>
<tr>
<td>Quizzes</td>
<td>30%</td>
</tr>
<tr>
<td>Labs</td>
<td>10%</td>
</tr>
</tbody>
</table>

Final Grades are obtained by converting the numerical scores using the conversion table below. The dividing line between grades may be adjusted by up to 3% to account for natural breaks in the numeric scores. Further, the "out of" values for the assignments may be adjusted downward to account for difficulty. Labs are graded on attendance and participation.

The grading scale will be:
90 – 100 A  
80 – 89  B  
65-79  C  
50-64  D  
0-49  F

**Tentative Schedule**

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Formal and natural languages, intro to Linux, black box modeling</td>
</tr>
<tr>
<td>2</td>
<td>Variables, assignment, printing, mental models for computers and the internet</td>
</tr>
<tr>
<td>3</td>
<td>Operators, math, random numbers, algorithms</td>
</tr>
<tr>
<td>4</td>
<td>Conditionals, if statements, switch statements, trees, composing conditionals</td>
</tr>
<tr>
<td>5</td>
<td>Looping, while do loops, do while loops, flow charts</td>
</tr>
<tr>
<td>6</td>
<td>Combining branching looping and I/O</td>
</tr>
<tr>
<td>7</td>
<td>Functions</td>
</tr>
<tr>
<td>8</td>
<td>Code refactoring, Code profiling, Java Libraries, Objects, Exceptions</td>
</tr>
<tr>
<td>9</td>
<td>File I/O, for loops</td>
</tr>
<tr>
<td>10</td>
<td>Caesar Cypher, nested loops</td>
</tr>
<tr>
<td>11</td>
<td>Arrays</td>
</tr>
<tr>
<td>12</td>
<td>Records</td>
</tr>
<tr>
<td>13</td>
<td>Object Creation and Design</td>
</tr>
<tr>
<td>14</td>
<td>Object Creation and Design</td>
</tr>
</tbody>
</table>

**Tentative Lab Schedule**

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro to Linux</td>
</tr>
<tr>
<td>2</td>
<td>Variables, assignment, printing</td>
</tr>
<tr>
<td>3</td>
<td>Algorithms focusing on operators and math</td>
</tr>
<tr>
<td>4</td>
<td>Programming with conditionals, if statements, switch statements, trees, composing conditionals</td>
</tr>
<tr>
<td>5</td>
<td>Programming with loops: while do loops, do while loops</td>
</tr>
</tbody>
</table>
Combining branching looping and I/O
Programming Functions
Code refactoring, Code profiling, Java Libraries, Objects, Exceptions
Programming with file I/O and for loops
Programming with nested loops
Programming with Arrays
Programming with records
Object Creation and Design
Object Creation and Design

Note that there are no large exams in the class. Quizzes will be given out in lab (14 quizzes – one per week) or in class/online (11 quizzes total, spread through the semester). You are expected to attend all labs, and the lab portion of your grade will be determined based on your attendance and participation in labs. Assignments will be given continuously throughout the semester, with due dates ranging from a few days to a couple of weeks apart, depending on the complexity of the individual assignment.

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, in Cain Hall, Room B118, or call 845-1637. For additional information visit http://disability.tamu.edu

Academic Integrity
The Texas A&M University Official Honor Code says, “An Aggie does not lie, cheat or steal, or tolerate those who do.” It is no excuse to say that you did not know the Honor Code, and everyone commits to adhering to that code when they come to Texas A&M, upon enrollment. For additional information please visit: http://aggiehonor.tamu.edu

Attendance
The book gives a guide to the material covered, but the instructor will introduce additional material and viewpoints. If you must miss a class, you should obtain notes from a student who was present. In general, lecture notes will NOT be posted. Texas A&M University considers class attendance to be a matter of personal responsibility on the part of each student. University rules related to excused and unexcused absences are located on-line at http://student-rules.tamu.edu/rule07.

Excused Absences for Religious Holidays and Make-up Quizzes:
Texas House Bill 256 (effective 9/1/03) states “An institution of higher education shall excuse a student from attending classes or other required activities, including examinations, for the observance of a religious holy day, including travel for that purpose. A student whose absence is excused under this subsection may not be penalized for that absence and shall be allowed to take an examination or complete an assignment from which the student is excused within a reasonable time after the absence.” The Dean of Faculties has provided a list of days of religious observance for the semester, and quizzes have not been scheduled on those days. However, should there be a conflict for this reason, you will be allowed to make up the quiz.

Beyond this, makeup quizzes will be given for any excused absences, but not for unexcused absences.
Texas A&M University
Core Curriculum Cover Sheet

Initial Request for a course to be considered for the Fall 2015 Core Curriculum

1. This request is submitted by (department name): Computer Science and Engineering

2. Course prefix and number: CSCE 121
   Introduction to Program Design and Concepts

3. Texas Common Course Number: n/a

4. Complete course title: Introduction to Program Design and Concepts

5. Semester credit hours: 4

6. This request is for consideration in the following Foundational Component Area:
   ☐ Communication
   ☐ Mathematics
   ☐ Life and Physical Sciences
   ☐ Language, Philosophy and Culture
   ☐ Creative Arts
   ☐ American History
   ☐ Government/Poliitical Science
   ☐ Social and Behavioral Sciences

7. This course should also be considered for International and Cultural Diversity (ICD) designation:
   ☐ Yes   ☐ No

8. How frequently will the class be offered? Fall, Spring, Summer

9. Number of class sections per semester: 10-20

10. Number of students per semester: 200-400

11. Historic annual enrollment for the last three years: 409 ('11-'12) 427 ('12-'13) 463 ('13-'14)

   This completed form must be attached to a course syllabus that sufficiently and specifically details the appropriate core objectives through multiple lectures, outside activities, assignments, etc. **Representative from department submitting request should be in attendance when considered by the Core Curriculum Council.**

12. Submitted by:

    Course Instructor:  
    Date: 8/26/14

    Approvals:

    Department Head:  
    Date: 8/26/14

    College Dean/Designee:  
    Date: 8/26/2014

For additional information regarding core curriculum, visit the Texas Higher Education Coordinating Board website at [www.thecb.state.tx.us/corecurriculum2014](http://www.thecb.state.tx.us/corecurriculum2014)

See form instructions for submission/approval process.
In the box below, describe how this course meets the Foundational Component Area description for Social and Behavioral Sciences. Courses in this category focus on the application of empirical and scientific methods that contribute to the understanding of what makes us human. Courses involve the exploration of behavior and interactions among individuals, groups, institutions, and events, examining their impact on the individual, society, and culture.

The proposed course must contain all elements of the Foundational Component Area. How does the proposed course specifically address the Foundational Component Area definition above?

Computing has become pervasive throughout society. Humans touch computing on a daily basis and increasingly rely on computing technology in every aspect of their lives. Computers have revolutionized the ways that people access and remember information, make decisions, interact with each other, and organize society. For people to thrive in this increasingly computer-centric world, it is important that they understand the basic ways in which computers work, interface with individuals, and can be controlled. In this course, students learn about the ways that computer software is designed and organized, the abilities and limitations of computer programs, how programs reflect communication between people and machines, how errors are found and dealt with, and how systematic approaches can be used to analyze the behavior of computer programs. While different populations interact with computers in different ways and thus will have different requirements for how computers are used, this course will expose many of the fundamental aspects of computer technology critical to having a deep understanding of how computers work.

Core Objectives

Describe how the proposed course develops the required core objectives below by indicating how each learning objective will be addressed, what specific strategies will be used for each objective and how student learning of each objective will be evaluated.

The proposed course is required to contain each element of the Core Objective.

Critical Thinking (to include creative thinking, innovation, inquiry, and analysis, evaluation and synthesis of information):

Software forms the basis for the many of the ways individuals, communities, and societies interact today. A major part of this course deals with helping students learn to design and develop such software. Software development is primarily a problem-solving activity, in which developers create new and innovative solutions to well-formed problems within the constraints of the computer technology and the particular programming language being used. The development process involves understanding the various requirements for the software, design of a solution, implementation, testing and analysis of that implementation, and evaluation of the software for solving the given problem. In assignments for the class (both coding assignments and non-coding ones), students are assessed as to how well their design and implementation is able to solve the given problem. This measures how well they are able to analyze the problem (analysis), create a new solution to the problem (creative thinking, innovation), and think of and use appropriate tests for the program, making modifications as needed (inquiry, evaluation, and synthesis of information).

Communication (to include effective development, interpretation and expression of ideas through written, oral and visual communication):

While software can help facilitate communication among people and organizations, the software development process itself involves communication in multiple ways. Code is itself a form of (primarily written)
communication between a programmer and the computer. Students in this class are also taught about program documentation, and how the code they write is a way of communicating to future programmers; they are assessed on how well they have structured and commented in code to clearly communicate intent. The course includes topics related to developing user interfaces, a form of visual communication. Students learn the process for putting together graphical user interfaces, and are assessed on how well they have done so. Oral communication is a critical part of the software development process. Laboratory work in the class includes students presenting and discussing the code they have developed with others (this is a weak form of the formal code review process used in professional practice).

Empirical and Quantitative Skills (to include the manipulation and analysis of numerical data or observable facts resulting in informed conclusions):

The ways that people use computers today have an either obvious or somewhat hidden reliance on manipulation and analysis of numerical data. Reflecting this, many of the examples and assignments used in the course have a numerical component to them, in which programs are developed that can process and work with numerical data to perform some task. Students are directly assessed on how well they can do this. Furthermore, the testing and evaluation stages of the software development process involve examining the results of output to evaluate how well the software is meeting its desired goals, and possible revisions to address errors discovered in the process (an example of using both numerical data and observable facts to draw informed conclusions).

Social Responsibility (to include intercultural competence, knowledge of civic responsibility, and the ability to engage effectively in regional, national, and global communities):

Computers help facilitate interaction at all levels of society, from small groups through the global community. The course gives students familiarity with the basic concepts critical to the software that provides this interaction, and thus enables students to engage more effectively with their communities. The course includes material on software security, how errors can occur and the impact they can have, and how software can be designed so that errors that may still occur will have minimal societal impact. Assessment includes tests in which students are evaluated on their understanding of these issues.

Please be aware that instructors should be prepared to submit samples/examples of student work as part of the future course recertification process.
SYLLABUS

CSCE 121 – Introduction to Program Design and Concepts
Fall 2015

LECTURE: MWF 9:10-10:00 ZACH 102

LABS: Section 501 MW 8:00-8:50 RDMC 111H
Section 502 MW 10:20-11:10 RDMC 111H
Section 503 MW 11:30-12:20 RDMC 111A
Section 504 MW 1:50-2:40 RDMC 111A
Section 505 MW 3:00-3:50 RDMC 111A
Section 506 TR 9:35-10:25 RDMC 111H
Section 507 TR 11:10-12:00 RDMC 111H
Section 508 TR 12:45-1:35 RDMC 111H
Section 509 TR 2:20-3:10 RDMC 111H
Section 510 TR 3:55-4:45 RDMC 111H
Section 511 TR 9:35-10:25 RDMC 111A
Section 512 TR 12:45-1:35 RDMC 111A
Section 513 MW 8:00-8:50 RDMC 111A

INSTRUCTOR: Dr. Walter Daugherity Telephone: 845-1308 Office Hrs: MW 10:30-11:30
daugher+CSCE121@tamu.edu Teague 318A and by appointment

TEACHING ASSISTANTS:
Chieh (Jay) Chou, jc491986@gmail.com, sections 507, 509, 512
Ben Fine, ineb@tamu.edu, section 508
Keishla Ortiz Lopez, keishla.ortiz4@tamu.edu, sections 504, 505, 511
Colton Riedel, crriedel@tamu.edu, sections 503, 506, 513
Zelun Wang, wzlxjtu@gmail.com, sections 501, 502, 510

COURSE WEB PAGE: http://courses.cse.tamu.edu/daugher/csce121/15fall (coming soon)
Check regularly; you are responsible for posted information.

PEER TEACHERS: See http://engineering.tamu.edu/cse/academics/peer-teachers/current-peer-teachers or go to
HRBB 219, Peer Teacher Central (note--room will change). In addition to the peer teacher(s) for your section, you
may also consult other peer teachers if they are not busy.

CHAT ROOM: An online course chat room will be set up at piazza.com which can be accessed from a mobile
device. You will receive a message to join the chat room soon. You are responsible for information posted to piazza.

OPERATING SYSTEM/COMPILERS: Unix/Linux with g++-4.9 and Microsoft Visual Studio C++ 2013. These are
free to you. The TA’s will introduce these to you in the first labs. The Linux environment is where homework
will be graded, but VS may be easier to use on your home computer, particularly without internet. Familiarity with
both is advantageous in obtaining summer jobs.

Catalog Description: Computation to enhance problem solving abilities; computational thinking; how people
communicate with computers; how computing affects society; design and implementation of algorithms; data types,
program control, iterations, functions, classes, and exceptions; understanding abstraction, modularity, code reuse,
debugging, maintenance, and other aspects of software development; development and execution of programs.

Prerequisites: Prior programming course (high school or college).

Required Clicker: Purchase an “i>clicker” and register it at iclicker.com with your UIN. This is required for graded in-class quizzes.


Optional Online Code Lab: [http://www.tcgo1.com](http://www.tcgo1.com) or [http://www.tcgo2.com](http://www.tcgo2.com) (more information later)

Learning Outcomes: This course addresses ABET outcomes a, c, e, and k. Upon completion of the course, students are expected to:

1. Be able to analyze a problem statement and design a computer program that can solve the problem for well defined input and output. This includes the ability to apply concepts from various paradigms such as object-oriented, procedural, and generic programming.
2. Implement simple computer programs in a high-level language (C++). This includes:
   a. The use of various data types including primitives, vectors, strings, and structs
   b. The use of control structures, functions, and exceptions
   c. The use object-oriented and generic techniques such as inheritance, polymorphism, encapsulation, and templates
3. Be able to test and evaluate program behavior to verify an implementation and discover errors.
4. Identify bugs and errors, assess the impact they can have on software, and apply techniques to minimizing, removing or otherwise handling them.
5. Design and implement programs as means for communication, including:
   a. Effective use of commenting, layout, naming, and structure;
   b. Creation of simple graphical user interfaces.
6. Recognize, interpret, and use basic algorithms such as those for searching, sorting, lists, trees, and maps.

These objectives will all be assessed using homework, quizzes and exams. Some objectives may be assessed by evaluating student group programming projects and written reports.

Method of Evaluation:

- Homework (Labs, Drills, Papers) 30%
- One-hour Exams (2) (10 pts each) 20%
- Comprehensive Final Exam 25%
- Project 20%
- Attendance and Class Participation 5%
- (Pop quizzes and lab quizzes)

Grade Scheme

90 - 100 A
80 - 89 B
70 - 79 C
60 - 69 D
< 60 F
GRADING POLICIES:
Attendance: Lecture and lab attendance is expected. Infrequent unavoidable absences are understood, but each student is responsible for any missed material. For excused absences, students will not be penalized. See Section 7 of the Student Rules for the excused absence policy. For acute illnesses of less than three days, both option A and option B of section 7.1.6.2 are acceptable in this course. For unexcused absences, a grade of zero will be assigned for missed work. University rules related to excused and unexcused absences are located on-line at http://student-rules.tamu.edu/rule07.

Class Participation: Class participation will consist of attendance (in lecture and lab) and responses to short quizzes during lecture and lab meetings about the reading assignments and the lecture.

Lab Work: Submit lab assignments through CSNET (the CSCE departmental electronic turn-in system) and printouts to your assigned Teaching Assistant (TA). A text file (.txt) may be required for answers to non-program questions, explanations, or data. Each student should maintain backup copies of all work. Some lab quizzes will consist of short programming tasks, which will be performed during the lab and submitted to the TA for grading.

Late Work: Homework assignments which are submitted up to 48 hours late will be penalized 20%. Assignments more than 48 hours late will not be accepted without specific approval from the instructor. Labs submitted by web, e-mail, or any form other than through CSNET, unless approved in advance by your TA or the instructor, will not be accepted. Penalties will be waived for university excused absences provided the instructor is notified as stated in student rule 7.

Exams: Examinations will require the use of 8½ by 11 inch Scantron forms from Data and Research Services; you must purchase your own forms prior to each test! Missed exams will be rescheduled without penalty for an excused absence, or with a 20% penalty if the absence is not excused.

Project: A group project will be assigned to groups of 3 to 4 students each. Assignments to groups will be made by the instructor.

Mid-term grades: Midterm grades will be assigned to all students in week 8, and reported to the Office of the Registrar for students who have completed less that 30 hours of college work. You must keep track of your own grades from the papers handed back.

Quibbles: Grades may be appealed to the instructor using a “quibble form” available on the course web site. Note: All grade appeals must be made no later than two weeks after the grade is assigned.

Tentative Schedule:

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction; Objects, Types, and Values</td>
<td>Ch. 0-3</td>
</tr>
<tr>
<td>2</td>
<td>Computation; Errors</td>
<td>Ch. 4-5</td>
</tr>
<tr>
<td>3</td>
<td>Functions, etc.</td>
<td>Ch. 8, 6.4-6.7</td>
</tr>
<tr>
<td>4</td>
<td>Classes, etc.</td>
<td>Ch. 8, 6.3, 6.8</td>
</tr>
<tr>
<td>5-6</td>
<td>Input and Output; Midterm Exam 1</td>
<td>Ch. 10-11</td>
</tr>
<tr>
<td>7</td>
<td>A Display Model; Graphics Classes</td>
<td>Ch. 12-13</td>
</tr>
<tr>
<td>8</td>
<td>Graphics Class Design</td>
<td>Ch. 14</td>
</tr>
<tr>
<td>9</td>
<td>GUIs</td>
<td>Ch. 16</td>
</tr>
<tr>
<td>10</td>
<td>Vectors and Free Store</td>
<td>Ch. 17</td>
</tr>
<tr>
<td>11</td>
<td>Vectors and Arrays; Midterm Exam 2</td>
<td>Ch. 18</td>
</tr>
<tr>
<td>12</td>
<td>Vectors, Templates, and Exceptions</td>
<td>Ch. 19</td>
</tr>
<tr>
<td>13-14</td>
<td>Containers, Iterators, Algorithms and Maps</td>
<td>Ch. 20-21</td>
</tr>
</tbody>
</table>

Lab assignments and activities will follow the topics discussed in class. Homework assignments will be due regularly throughout the semester, with the specific due dates reflecting the complexity of the assignment. Assignments will be due every 1-2 weeks. The group project is expected to be assigned in week 10, and no other assignments will be given during the duration of that project.
American with Disabilities Act (ADA) Statement: The Americans with Disabilities Act (ADA) is a federal antidiscrimination 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, in Cain Hall, Room B118, or call 845-1637. For additional information visit http://disability.tamu.edu.

STUDENT RULES: You are responsible for complying with all provisions of the student rules posted at http://studentrules.tamu.edu.

Academic Integrity Statements

SCHOLASTIC DISHONESTY: Scholastic dishonesty will not be tolerated. Working together on homework assignments is encouraged, but the final product submitted for grade must be the individual work of the person turning it in. In other words, it is all right to discuss and to assist each other concerning programming strategy or technique for one student to help another debug code which will not work; but each student is expected to write his or her own programs from beginning to end. In this regard, if code from two or more students is essentially identical, and it is determined to the satisfaction of the instructor that the code is not the product of the individual, all students involved are subject to the Texas A&M University Honor System Rules, including a course grade of F* (* = for academic dishonesty) if this is the first offense, plus additional penalties as determined by the Aggie Honor System Office if this is not the first offense.

It is imperative that each student clearly understand those rules and the serious consequences that can result from adjudication of an Honor Code Violation. In particular, every student should understand that complicity – helping or attempting to help another student commit an act of academic dishonesty also constitutes academic dishonesty and carries the same punishment as cheating. In other words, if you provide your solution to another student, who turns it in for credit, you are both subject to the same consequences.

Plagiarism is the presentation of the work of someone else without giving him or her due credit. You can copy the words of others as long as you clearly identify them as such. In fact, documented use of program libraries is encouraged. Submitted work will be examined for plagiarism using computer software designed for that purpose. Examinations are meant to measure the knowledge or skill of each individual, so giving or receiving unauthorized assistance during tests and quizzes is cheating. It is assumed that college students know what is honest and what is not.

AGGIE HONOR CODE: “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.

The following statement must be typed on every lab, project, or exam submitted for grading in this course. Any papers that do not include a signed statement like the one below will not be graded.

“On my honor, as an Aggie, I have neither given nor received any unauthorized aid on this academic work.”

OR, for assignments with multiple components:

“On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment.”

_____________________________  ________________________________
Typed or printed name of student  Signature of student