Course Objectives:

CS 159 introduces the tools of software development that have become essential for creative problem solving in Engineering. Educators and employers agree that it is important for future Engineering professionals to be able to function as part of a technical team and develop the ability to communicate advanced technical concepts. CS 159 will require students to work in assigned teams for all lab assignments. Educational research informs us that structured collaboration leads to increased learning gains for students participating in an introductory programming course.

- **Collaboration is a requirement of the course.** You will be assigned to your teams by course staff.

CS 159 explores programming concepts in a language using computing environments that are new to most students. Our goals are for you to recognize how programming concepts are common across programming languages and how those concepts can be used to solve a problem.

Pre-requisites and Preparation:

The University expects students to place 4-8 hours outside of class per week in preparation for each credit hour of an eight-week summer course. The key to success in this course requires preparation, taking initiative to begin assignments early, and regular review of course materials. **The most successful students in previous distance learning offerings of CS 159 report habits such as reading the text, reviewing every on-line demonstration, and daily code writing that goes beyond the minimum of completing assignments.**

- **CS 159 does have a co-requisite of ENGR 131.** Authorized equivalent courses or consent of instructor may be used in satisfying the course co-requisite.

- **What does the co-requisite imply?** ENGR 131 as a co-requisite would imply that you are or have been (1) enrolled in a course that will expose you to the fundamentals of MATLAB or (2) have had some previous programming experience in a language such as C, C++, or JAVA. Most students in CS 159 will be enrolled in or have already completed ENGR 132 in which they are/were exposed to writing programs in MATLAB using structured programming concepts.

Course Staff:

**Instructor:** William Crum  
**Office Location:** HAAS G-26

**Graduate TA:** Ryan Senkpeil  
**Office Location:** HAAS G-25

**E-mail Address:** cs159staff@lists.purdue.edu this address sends an e-mail to both the instructor and graduate TA and will enable a faster response. Please be sure to only use your Purdue address when e-mailing course staff.

Important Dates:

<table>
<thead>
<tr>
<th>Midterm Exam #1</th>
<th>Midterm Exam #2</th>
<th>Final Exam</th>
<th>Academic Calendar</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 30, 2016 12:00pm – July 1, 2016 8:00pm</td>
<td>July 19, 2016 12:00pm – July 20, 2016 8:00pm</td>
<td>August 4, 2016 12:00pm – August 5, 2016 8:00pm</td>
<td>Last Day to Drop: July 13, 2016</td>
</tr>
</tbody>
</table>

Blackboard (**mycourses.purdue.edu**):

All relevant class information, updates, and announcements will be available on the Blackboard site.

In the event of a major campus emergency, course requirements, deadlines, and grading percentages are subject to changes that may be necessitated by a revised semester calendar or other circumstances.

- In case of campus emergency, check Blackboard, do not email or call course staff.
- The instructor reserves the right to revise the syllabus and will provide notice.
Course Required Materials:

- Programming Applications for Engineers Course Packet (Spring 2016 edition)

Grading:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>35</td>
</tr>
<tr>
<td>Lab Tasks</td>
<td>60</td>
</tr>
<tr>
<td>Midterm Exams</td>
<td>150</td>
</tr>
<tr>
<td>Final Exam</td>
<td>100</td>
</tr>
<tr>
<td>Pre-Lab Quizzes</td>
<td>60</td>
</tr>
<tr>
<td>Post-Lab Quizzes</td>
<td>60</td>
</tr>
<tr>
<td><strong>Total Possible:</strong></td>
<td><strong>465</strong></td>
</tr>
</tbody>
</table>

Grades:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Points Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>395</td>
</tr>
<tr>
<td>B</td>
<td>350</td>
</tr>
<tr>
<td>C</td>
<td>305</td>
</tr>
<tr>
<td>D</td>
<td>260</td>
</tr>
</tbody>
</table>

An equivalent number of points to earn a C are needed to receive a Pass if taking the course Pass/No Pass. It is recommended that graduate students in the course elect this option (see your advisor).

- The instructor reserves the right to lower the minimum score required for each letter grade. If such a move is made it will not be announced until after the final exam.
- At no time during the semester will it be speculated if this will be done or how much any given cutoff will be lowered. You should have no expectation that all cutoffs if moved will be moved by an equal amount.

Assignments:

**Pre-Lab Quizzes** (12 total, 5 points each):

Pre-lab quizzes are based on problem sets that will require students to review the text and on-line demonstrations for the most important terms and concepts as we progress through the material this semester. A quiz on this specific material must be completed at least 48-hours prior to the due date of the corresponding lab programming assignment. Each quiz will be found on Blackboard in the Labs and Quizzes folder.

**Lab Assignments** (12 total, 5 points each):

Lab assignments are to be completed collaboratively in your assigned lab groups and each of these lab programming assignments will be due at 11:00pm on Wednesdays and Fridays this semester.

- Collaborative groups are expected to communicate who will submit the assignment, when the assignment will be submitted, when updates on progress can be expected, and how successful submission will be confirmed with all participating group members.
- Lab partners failing to participate and contribute to the satisfaction of all group members will not receive credit for the lab assignment.

**Lab Quizzes** (12 total, 5 points each):

Due at the same time of each lab programming assignment will be an individual assessment of your knowledge related to the current topics of the course and recent programming assignments. Knowledge of course standards and good programming practices will be evaluated throughout the semester.

- The best way to prepare for lab quizzes is to actively participate with your team including contributing to the development and implementation of the programming problem solution.
Lab Schedule:

<table>
<thead>
<tr>
<th>Week of</th>
<th>Monday</th>
<th>Wednesday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>June-13</td>
<td></td>
<td>Pre-Lab #1</td>
<td>Lab #1</td>
</tr>
<tr>
<td>June-20</td>
<td>Pre-Lab #2</td>
<td>Lab #2</td>
<td>Lab #3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-Lab #3</td>
<td></td>
</tr>
<tr>
<td>June-27</td>
<td>Pre-Lab #4</td>
<td>Lab #4</td>
<td></td>
</tr>
<tr>
<td>July-4</td>
<td>Pre-Lab #5</td>
<td>Lab #5</td>
<td>Lab #6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-Lab #6</td>
<td></td>
</tr>
<tr>
<td>July-11</td>
<td>Pre-Lab #7</td>
<td>Lab #7</td>
<td>Lab #8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-Lab #8</td>
<td></td>
</tr>
<tr>
<td>July-18</td>
<td></td>
<td>Pre-Lab #9</td>
<td>Lab #9</td>
</tr>
<tr>
<td>July-25</td>
<td>Pre-Lab #10</td>
<td>Lab #10</td>
<td>Lab #11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-Lab #11</td>
<td></td>
</tr>
<tr>
<td>August-1</td>
<td>Pre-Lab #12</td>
<td>Lab #12</td>
<td></td>
</tr>
</tbody>
</table>

Notes on all Blackboard quizzes this semester:

1. You will receive only a single attempt to complete a quiz.
2. The 10-minute time limit will begin when you start the quiz. The time remaining cannot be paused and the quiz resumed at another time. If you exit Blackboard you will need to log in again and may resume the quiz with that time which remains.
3. Each question can be answered only a single time. Once you move to the next question you cannot return to revisit a previous question.
4. You will be able to view your score when you complete the quiz but you must wait until after the deadline for the quiz is passed before your effort can be reviewed on Blackboard. Once the quiz is released for review you can use the following steps: [http://web.ics.purdue.edu/~wcrum/cs159/reviewLabQuiz.jpg](http://web.ics.purdue.edu/~wcrum/cs159/reviewLabQuiz.jpg)

Homework Assignments (7 total, 5 points each):

The homework assignments are **individual** efforts designed to give you the opportunity to solve problems on your own **without the assistance of other students**.

All assignments will be posted on Blackboard 5-7 days before they are due.

Please review the course policies as they relate to **academic integrity** found later in this document.

<table>
<thead>
<tr>
<th>Homework Assignment</th>
<th>Due at 11pm on</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>June-20</td>
</tr>
<tr>
<td>2</td>
<td>June-27</td>
</tr>
<tr>
<td>3</td>
<td>July-4</td>
</tr>
<tr>
<td>4</td>
<td>July-11</td>
</tr>
<tr>
<td>5</td>
<td>July-18</td>
</tr>
<tr>
<td>6</td>
<td>July-25</td>
</tr>
<tr>
<td>7</td>
<td>August-1</td>
</tr>
</tbody>
</table>

Our expectation of your lab instructor is that he/she grades your assignment in a **timely manner and provides you with adequate feedback** for improvement. If you feel this is not the case please address your concern to your grader and the staff of the course. Typically, your grader should be returning assignments 2-3 days after a given assignment is due.
Exams (two midterms 75 points each, one final exam 100 points):

Exams will be individual assessments of your knowledge. Exams will consist of multiple-choice problems covering programming concepts, best programming practices, lab and homework assignments, and the interpretation of code.

All exams will be administered through Blackboard.

- The length of each midterm will be 60 minutes and the final exam will have a two-hour limit.
- An exam can only be attempted one time and your time limit begins once you start the exam.
- Problems can be revisited during the time of the exam.

You are required to take your exam under the supervision of an exam proctor which must be approved by course staff. More information on the requirements and application for a proctor can be found on-line:
http://web.ics.purdue.edu/~wcrum/cs159/summer16/proctor.pdf

Students on, or able to travel to, the West Lafayette campus will use course staff members as proctors.

- Resources are permitted during the exam including your book, notes, and programs you have previously written. Referencing another individual during the exam is not permitted.
- Any suspicion of academic integrity concerns will result in completing future exams only under the supervision of course staff while on the West Lafayette campus.

Re-grade Request Policy:

To request a re-grade on any assignment you must make your request to the (CS159staff@lists.purdue.edu) e-mail address. You have five days to appeal any grade from the day the assignment is returned to you. After that period the grades are frozen and no appeal will be considered.

A re-grade request must include the following:

1. The original graded assignment that was returned to you (if applicable).
2. The reason you believe a re-grade is warranted.

Note: A re-grade is not a second chance to complete an assignment. It is not a means to challenge posted policies, such as the acceptance of late work or course standards.

Absences:

Only documented serious hardships will be considered for any make-up work.

If you have documentation of what you consider to be such a hardship then you must contact the course staff in a timely manner to the (CS159staff@lists.purdue.edu) e-mail address when you are able to resume participating in the class. Any student who knows in advance of an absence must make a request for consideration one week prior to the planned absence.

- University policies on absence and absence reporting are available from the Office of the Dean of Students: http://www.purdue.edu/studentregulations/regulations_procedures/classes.html
- Do not expect assignment deadlines or exam dates to be altered for reason of personal travel.
- Make-up requests for reasons of illness MUST be accompanied by a physician’s note stating the dates you were under their professional care and the date you were cleared to return to school/work.

Important Assignment Guidelines:

All assignments must abide by the programming and documentation standards of the course, which are found later in this document. In ALL cases no credit will be given for programs that do not compile. Programs that execute and meet minimum assignment requirements but are not logically correct or complete may be considered for partial credit.

To receive full credit, your program must produce correct results, be well-designed, be efficient, follow assignment requirements, and adhere to course programming and documentation standards.

An assignment that is not submitted as expected cannot be considered for a grade. Only work submitted correctly prior to the assignment deadline can be considered for grading. Late work is not accepted.
Resources and course staff may become heavily loaded as an assignment deadline nears. Waiting until the last minute to work on your programming assignments is discouraged! Course policy is NOT to extend deadlines unless campus resources (not your local ISP) are unavailable for an extended period near the deadline for an assignment.

You are responsible for knowing how to use the technology utilized by the course, this includes but is not limited to UNIX and related course tools such as the assignment submission script.

Be sure your account is set-up correctly as introduced in the first lab at the start of the semester.

**Plan to submit work early! Allow sufficient time to seek assistance should you experience any difficulties with assignments or submitting an assignment.**

**Academic Integrity:**

A very detailed set of criteria that is enforced rigorously related to academic integrity is applicable to CS 159. The consequences for violating course policies are serious.

You are encouraged to discuss any CS 159 topic including ideas about how to complete assignments. But, under **no circumstances will exchange of code via written or electronic means be permitted** between individuals for individual assignments or teams for collaborative assignments.

**It is considered dishonest either to read another team's solution or to provide anyone with access to your work (or that of another student).** Be very careful when working with others on individual assignments as this is generally discouraged. The work you submit must be your own original effort and not the result of unacceptable, even if unintentional, collaboration.

Every student is responsible for protecting their own work. Do not make the assumption that roommates, neighbors, significant others, or other trusted individuals would not take advantage of knowing your password, having access to your computer (use a password protected screen saver, log out when done), or taking a picture of your work when left on an unattended machine. **You are responsible for such events that leave your work unprotected.**

Do not make the mistake of thinking that **superficial changes in a program** (such as altering comments, changing variable names, interchanging statements, or additional white spaces) will avoid detection. If you are unable to complete the work yourself, it is unlikely that you will succeed in disguising the work of another as your own. We are adamant that **violations in any form will not be tolerated.** Even the most trivial assignment is better not done than if you violate course integrity policies to complete it.

**Why enforce academic integrity?** Academic integrity violations amount to theft. Theft of the work of the individual who developed the assignment, theft of that individual's time, and theft of the instructor's time to conduct an inquiry into such matters. It also amounts to **theft from every student** who has come to Purdue University, made a significant financial investment in their education, and has the expectation that their degree will be valued by employers and other academic institutions when they leave.

**When is it no longer acceptable to discuss an assignment with another student or someone not from my group?**

- Discussions with peers are most appropriate during the early phases of solution development. Once you begin to implement your solution or have constructed detailed flowcharts or structure charts you should be referencing course staff members exclusively for assistance.

- Working closely with another student on a homework assignment may result in highly similar work due to collaboration. Collaboration may not have been the intended approach to solving the problem but the end result of working closely with others for extended periods of time.

As easy as it is to share an electronic copy of a file, to gain access to a file through account sharing, sharing a hard copy of your work, or seeking assistance from strangers on the Internet, it is as just as easy to analyze and detect such sharing as it results in similar efforts being submitted.
Assume that every submission you make during the semester will be analyzed by a software similarity service. This service will return the percentage of similarity between your solution and those submitted by others in the course. Additionally, the service indicates the number of lines matched among submissions. **You will be solving problems this semester that have no unique solution and your solution is expected to be uniquely yours.** Concerns regarding any of our policies should be addressed during office hours prior to the deadline of an assignment.

The software service utilized is not for profit. The service does not retain your file. The course will retain your files for the purpose of record keeping for the current semester and may retain your files for similarity comparisons in future semesters.

**Minimum consequences for violating course policies will include:**

- First offense, a zero for the assignment, a reduction of one letter grade at the end of the semester, AND a referral to the Office of the Dean of Students for disciplinary action.
- Second offense, a zero for the assignment, a failing grade for the course, AND a referral to the Office of the Dean of Students for disciplinary action.
  - Penalties for being found in violation from the Office of the Dean of Students will typically range from disciplinary probation, to probated suspension, to being dismissed from the University.

**Exceptions to the minimum consequences:**

- Any violation on an exam will result in a failing grade for the course and a Dean of Students referral.
- Acts such as misrepresentation of identity (or location) will result in a failing grade for the course and a Dean of Students referral. **This includes participating in a quiz for another student.**

**Collaborative Learning/Teaming/Participating as a Member of a Technical Team:**

Here are our expectations of you and your group:

1. **Make time to meet with your group regularly.** There are 168 hours in a week, finding some common time for two or three people to meet should not be difficult. It is acceptable for just part of the group to meet some of the time if everyone cannot attend every meeting. It is the responsibility of each individual to plan their contribution to the group effort accordingly.
   - A group may exclude the name of a member from the lab assignment as a means of indicating a lack of satisfactory contribution to the group effort.
   - **How does a group meet on-line?** Consider tools such as Google Docs, GroupMe, or other collaborative document or code development services.

2. **Allow everyone an opportunity to express their ideas** on how to approach an assignment. One benefit of collaborative teaming is that everyone brings a different set of skills to the group and the resulting effort often is a stronger one than if it was completed individually.
   - When a group member becomes unresponsive to requests to meet or fails to update the other members of the group then those contributing members must continue without unresponsive member.
   - **How is communication a challenge in a distance learning format?** Be sure the members of the group share their availability with the other members of the team so that each member knows when they can expect a response to inquiries and concerns. A shared document (Google Docs) might be one way to track progress on an assignment to ensure that the group finishes all agreed-upon tasks on time.

3. **All group members must be satisfied with the final submission.** It is not acceptable for a group to submit an assignment that is not approved by all group members. "It is good enough" may be true for you but it is unfair for the others in the group who aspire for the strongest grade possible.
   - Likewise, each group member must be satisfied with your contribution to the group effort.
   - **How will we know who was a contributor to the final submission?** If the Purdue University e-mail address of each member is correctly placed in the header of a lab assignment then each member will receive a confirmation message when a submission is made. This confirmation message will come via e-mail and include the work that was submitted.
4. Each group member must fully understand the entire assignment submitted. Do not start your group discussions by trying to delegate the tasks to the different group members. Everyone must understand and contribute to every aspect of the assignment and its development.
   ◯ Assignments are an opportunity for you to demonstrate your knowledge of the programming concepts being utilized. Additionally, an assignment can be used to serve as a measure of what you don't know.
   ◯ Is there really a problem with delegating whole assignments to individual members? All members will have to take a lab quiz relevant to the current assignment. Failing to understand the assignment may result in a lower quiz score. Each assignment also represents a topic in the course and may be your only opportunity to work with that concept prior to an exam.

5. Designate who will turn in the assignment, when it will be turned in, and how successful submission will be communicated with the rest of the group. Only one person from the group will submit the assignment. Set a goal to submit the assignment well in advance of the due date to avoid any last minute problems related to group communication.
   ◯ All groups are encouraged to exchange that work in-progress. This can be accomplished by making a submission with the Purdue e-mail address of each group member in the assignment header or through the use of a collaborative document (Google Docs). Should a group member be unresponsive the remaining group members can proceed without starting over.

6. You will work with the group assigned. There is no other option in this course. Please contact course staff with concerns you may have with your group. Take a professional approach with your group as would be expected of you at an internship or co-op experience.
   ◯ Groups may be re-assigned as warranted by changes in enrollment.
   ◯ Future group assignments may take into consideration assignment completion and participation in on-line forums as an indicator of your interest in participating in the course. Active students in the course should not be burdened with partners who are not willing to stay current with course content.

Teaming experience is important to employers!

Consider how your experience on a technical team in the lab might help you answer the following common questions asked in interviews:

- Tell me about a well-functioning team that you were on. Why do you think the team worked so well together?
- What actions and support, in your experience, make a team function successfully?
- Have you been a member of a team that struggled or failed? What assessment did you make of the reasons for the failure?
- Give me an example of a time when you had to teach a skill to other engineers.
- Some of the best ideas are born out of an individual’s ability to challenge others’ ways of thinking. Tell me about a time when you were successful doing this.
- For what assistance do fellow team members turn to you?
COURSE PROGRAMMING AND DOCUMENTATION STANDARDS

Programming Standards – How we expect you to approach the development of your solution. Includes our definitions of what are good programming practices and what we consider to be bad practices.

Documentation Standards – How we want your code to be formatted and documented when submitted.

Please read and revisit the section carefully throughout the semester.
Documentation, Programming, and Course Standards

Grading Standards

Every assignment will attempt to provide sufficient detail on all requirements. However, because the problems solved in this class have no unique solution, we will rely on the general guidelines listed below when determining the appropriate evaluation of your work.

You will always be evaluated on the following aspects of your assignments:

1. Following provided assignment requirements.
2. Developing the appropriate logic to solve the problem, accepting input in the order expected, and producing expected output.
3. Implementation of good programming practices in the final algorithm.
4. Your solution should be efficiently designed to minimize wasting the limited resources of the computer.
5. Abiding by course documentation standards.

It is possible to receive less than full-credit for an assignment if each of the above is not followed to course expectations. Aspects of your submission other than producing correct output are important and can result in a loss of significant points on an assignment.

Documentation Standards

The requirement of good documentation practices is universal among professionals; the primary motivation of having documentation standards is to make your code and logic as easy to understand and for your grader to evaluate as possible. The solutions you are in the process of developing must be easily understood when seeking assistance both in this class and from your colleagues and supervisors in the future.

These standards are not merely recommendations they are required for the course and failure to comply will result in a loss of points.

- The course notes will attempt to abide by these standards as closely as possible to serve as an example. However, to save space in the notes and time during lecture, headers and comments may be excluded; this is not an option for your assignments.
- Various textbooks (including the required textbook) do not abide by all of our standards; please refer to this document for clarification.

Points on an assignment will be deducted if your code and/or logic is difficult for your grader to evaluate because it fails to meet the expectations outlined in this document.

Is there real world relevance for programming standards? Most companies have extensive documentation requirements in order to facilitate the transfer of previously developed code from one employee to another without the need to spend a great deal of time trying to understand the work implemented by the original developer.
Assignment Headers

- Assignment headers are used in the course to explain who the author/s is/are and provides a detailed description of the logic being implemented in a program or user-defined function.
- There are five total headers we will use this semester. Two for labs, another two for homework, and one for user-defined functions within assignments. Two are for Octave/MATLAB assignments and the remaining three for C assignments.
- The use of a header MUST NOT interfere with the compilation or execution of your program.

Lab Header Example (for C programs)

```c
/***************************************************************************
* Programmers and Purdue Email Addresses:
* 1. login1@purdue.edu
* 2. login2@purdue.edu
* 3. login3@purdue.edu (delete line if no third partner)
* Lab #:
* Academic Integrity Statement:
* We have not used source code obtained from
* any other unauthorized source, either modified
* or unmodified. Neither have we provided access
* to our code to another. The project we are submitting
* is our own original work.
* Lab Division and Section:
* Program Description:
***************************************************************************/
```

- Be sure to list the Purdue e-mail address for each participating group member! Grades are collected and uploaded to Blackboard based on your Purdue career account.
  - You MUST delete the login1, login2, login3 addresses before attempting a submission.
- After each submission of a lab assignment the members of the group who are correctly identified in the lab header by their Purdue e-mail address will receive a confirmation e-mail.

FAQ: Where are the header files in my career account?
- **Answer:** In the ~/CS159 directory of your career account.

FAQ: How do I restore the header files if they are deleted?
- **Answer:** Execute the following command: `copyHeaders`

FAQ: How do I know which headers are for MATLAB?
- **Answer:** `head_Mhw` and `head_Mlb` are for MATLAB assignments.
Homework Header (for C programs)

/*******************Homework Header (for C programs)***********/
/*
 * Programmer and Purdue Email Address:
 * login@purdue.edu
 *
 * Homework #:
 *
 * Academic Integrity Statement:
 * I have not used source code obtained from
 * any other unauthorized source, either modified
 * or unmodified. Neither have I provided access
 * to my code to another. The project I am submitting
 * is my own original work.
 *
 * Lab Division and Section:
 *
 * Program Description:
 */
/****************************************************************************/
Object Names / Identifiers

- We will consider the following to be objects in this course: **variables, functions, symbolic/defined constants.** You must **give your objects names that reflect their purpose in the program.**

  ```c
  int x;
  int y;
  int z;
  ```

  **Course Standard: Declare only one variable per line!**

- Can you determine what the variables x, y, and z accomplish for the program based only on the name of the variable? Perhaps they hold information on a 3D point? If the program contains no 3D coordinate data, then the use of x, y, and z as variable names is unclear.

- **Rarely is a single character identifier for an object meaningful.** A reasonable exception would be for a loop control variable used for simple counting or to access the elements of the array.

- The use of the underscore to begin an identifier is not a practice that we will use in the course.

  ```c
  int numStudents;
  float objectMass;
  char courseGrade;
  ```

  Relevant variable, constant, and function names assist in the **documentation, and readability,** of the logic you are attempting to implement.

Symbolic/Defined Constants

- **THE SELECTION OF IDENTIFIERS FOR SYMBOLIC/DEFINED CONSTANTS MUST BE IN ALL CAPS.**
- This standard allows any programmer to differentiate between a variable, a function call, and a constant in an expression.
- Define all symbolic/defined constants at the top of your program immediately following the assignment header.
- Because the use of symbolic/defined constants can improve the documentation of a program it is a standard of the course that you attempt to maximize your use of symbolic/defined constants and minimize your use of literal constants (see chapter 3).

**Note: Memory constants will not be used in this course.**

Commenting

In the C programming language we have two possible implementations of comments:

- The single line comment // that will instruct the compiler to ignore everything after // to the end of the line.
- The multi-line comment /* */ that will instruct the compiler to ignore everything between the /* and */. See the headers above for an example of this type of comment.
  - Multi-line comments cannot be nested in the C language.
- MATLAB utilizes % for a single line comment and % { and % } for a multi-line comment.

What **must** be commented?

- **EVERY variable** that you declare must have a brief comment to describe its purpose in the program!
  - In C, comment variables to the right of where they are declared.
  - In MATLAB, list and comment all variables in the relevant course assignment header.
- Within a given function it is expected that the occasional complex segment of code be documented in order to inform others regarding that logic you are attempting to implement.
  - For example: It is recommended that you place a line or two of comments before any significant selection or repetition construct.
Indenting

You must indent all code inside the body of a structure **two (additional) spaces**! Rarely should you ever begin a line of code in the first column of your editor!

Here are some common structures properly indented according to our standards:

```plaintext
if(x == 3 || y == 2)  
{
    z++;  
    a = a - 1;
}

for(x = 1; x <= MAX; x++)
{
    printf("%d", x);
}

//NESTED STRUCTURE BELOW, NOTICE ADDITIONAL INDENTING REQUIRED!!

if(y == 3)
{
    for(g = 0; g > MIN; g--)
    {
        y = y - g;
    }
}
```

**More Indenting Examples:**

<table>
<thead>
<tr>
<th>Correct</th>
<th>Incorrect</th>
<th>Incorrect</th>
</tr>
</thead>
</table>
| if(x == 3 || y == 2)  
{  
    z++;  
    a = a - 1;
}  | if(x == 3 || y == 2)  
{  
    z++;  
    a = a - 1;
}  | if(x == 3 || y == 2)  
{  
    z++;  
    a = a - 1;
}  |

Switch constructs will be indented as seen in the C text! See page 258.

White-Spacing

- Please place an additional line between sections to help "group" the code
- Balance the use of additional lines and failing to place any throughout your code.
  - There is no need to double space all lines of code.
- It is a requirement to **place a space between operators and operands**, such as: \( y = y - g; \)
- In MATLAB do not use the command `clc` to clear the screen.
Programming Standards

There are numerous reasons for programming standards; in this course the purpose of such standards as described here is to help you develop good habits for your future programming endeavors. For some of you, this requirement will require breaking previously developed bad habits.

What makes a habit bad? If a few of the implementations discussed below are not used properly they can expose your program to difficult errors for a beginner to debug, program crashes, hacking, or represent programming language features that will go obsolete in the near future.

Keep Local Declarations Separate from Executable Statements

- The current version of the gcc compiler permits variable definitions anywhere in a function. This aspect of the compiler does not follow the historical standard for the C programming language that all variables are declared prior to the start of the executable statements. You may not take advantage of this “feature”. Failing to keep this expectation may result in the grading of your program as if it did not compile.
  - See Figure 2-2 in Chapter 2 of C text for more information.

Scope

- VARIABLES ARE NEVER PERMITTED TO HAVE GLOBAL SCOPE.
- Do you need to share a variable with another function? Do you need a variable value to change within a function and to retain that change after the function terminates?
  - Then you must understand how to pass the variables/values to and return them from a user-defined function.
    We reserve the right to assign a score of zero for the implementation of global variables.
- We expect in this course that all function declarations will have a global scope.
- Do not reuse an identifier with two objects that have overlapping scope.

Symbolic/Defined Constants and other Pre-processor Directives

- All pre-processor directives must go at the top of your program just below the appropriate assignment header for the file.
- You can read much more about symbolic/defined constants in your C text (appendix G) but one purpose of a symbolic/defined constant is to make your role as programmer easier when the requirements for a program are modified in the future. You will also find these constants improve the documentation of your program.

The use of \{ and \}

- You may observe in the C book, but not in the notes, that it is possible in certain situations to omit the braces around the body of many structures such as if, for, and while.
  - Do not develop, or maintain, this habit! Beginners have a difficult enough time pairing up their braces and a misplaced brace is one of the more difficult errors to resolve.
- We will discuss the dangling else problem in chapter 5 (page 245 of the C text, figure 5-14) as an example of one difficult logical error to debug when you omit the braces as we have required. You will also encounter the book’s lack of \{ and \} when you try to make sense of the sorting algorithms coded in chapter 8 of the text.

For all relevant selection and repetition constructs in C the use of \{ and \} is required.

- MATLAB/Octave does not make use of \{ and \} to delimit structures such as functions, selection, and repetition. It is recommended that you appropriately indent, make use of ; when reasonable, and correctly implement the end statement to format your MATLAB/Octave code.
Proper Use of Repetition

- Use for loops only with counter-controlled processes.
- Make use of all three expressions with every for loop. Do not leave expressions empty or place meaningless statements in any of the expressions such as \( x = x \).
- The use of the comma operator is probably unnecessary in for loops for this class. See page 323 of the C text.
- The use of perpetual loops would violate the expectation that control-forcing statements are prohibited.
  - See next section.
- Recursion must only be used with counter-controlled processes.

Control Forcing Statements

- You may observe control forcing statements being used in the texts but their use is prohibited within the course. Often there are better ways to dictate the control of your program without using one of the following statements:
  - exit, goto, break (permitted only with switch structures in C), continue
  - Multiple return statements in the same function. A function, much like the flowcharts we will discuss in lecture, must have only one starting and ending point. Using multiple return statements enables multiple ending points for a single function (algorithm).

Standard Libraries and Functions

- You are permitted to use any of the functions in the standard C libraries introduced in class. It is your responsibility for asking a course staff member about a function or library we have not discussed in class that you would like to use.

User-Defined Functions

Once we introduce Functions (chapter 4) only the following will be permitted in the main function of a program:

1. Declaration of those variables that need to be passed between functions.
   - All variables in the function definition should have an explicit data type.
2. Functions called by main.
3. A minimum amount of control code (loops, selection) to maintain main as the control center of the program.

The main function is intended to be the main function of the program and a majority of the function calls should originate in main. Most of the data defined in the program should be done in main.

According to the definition of a function, each function must only have an individual (single) specific task. Failing to make good use of user-defined functions on relevant assignments will result in no partial credit.

For further discussion on user-defined functions read the following sections of your text:

- Chapter 4.8 Structure Charts
- Pay particular attention to the section on functional cohesion on page 211.

Passing individual values by address should meet the following criteria:

1. When more than one value needs to be revised and that revision be retained in the calling function once the called function terminates.
2. The function continues to meet the definition of being functionally cohesive such that it completes only a single fundamental task.
Declaration of Arrays

All arrays will be **static** (of fixed-length) until we introduce the topic of **dynamic** memory allocation in chapters 9 and 10 of the C programming text. The application of variable-length arrays as seen in the example on pages 478 – 479 **will violate course standards** as it permits the local declarations and executable statements in a function to overlap.

- It is unacceptable to make use of a variable to represent the size of an array in an array declaration.
- A symbolic/defined constant will be used when declaring arrays.

Advance Topics/Implementations

- In course assignments we will state from which chapters you may reference when developing your solution.
- We commonly view advance implementations (material not covered yet in lecture) in code as suspicious in that we consider such an implementation to be beyond what most students in the course are capable of and suspect that the author of the code may be someone not currently in the class.

Efficient Design

- One measure regarding the quality of ALL assignments is the implementation of efficient design. Engineering students should anticipate and appreciate that design is critical to the evaluation of any product including software.
- We require that you implement an efficient and logically correct algorithm as part of your assignment. In some cases this may make the distinction between letter grades on the assignment. To receive full credit on an assignment we expect **correct logic AND good design**.

Some guidelines related to the efficient use of the limited resources of the computer:

- Avoid wasting memory.
  - Do not use multiple arrays (or dimensions) when one will do the job.
  - Do not declare (in C) arrays to be large beyond your data needs (specified by the assignment).
  - Do not use arrays when they are unnecessary.
- Avoid wasting the CPU’s time.
  - Do not write nested loops for tasks that can be done in one (or none).
- Shorter code is usually better.
  - Recognize patterns in your code. Avoid duplication of the same logic within a program.
    - If you find yourself repeating code in several places, create a function such that the code only appears once in your program and is called when needed.
    - On the other hand, if you find that several functions have been written that do essentially the same thing, or contain a considerable amount of similar code, then attempt to consolidate as much as possible.
  - Use the top-down design.
    - Each function should have a well-defined task and not a series of tasks.
- Topics of efficient design will be commented on, where appropriate, in lecture.

Syntax Errors and Warnings

- Any assignment submitted with a **syntax error** will receive **no consideration for partial-credit**.
  - You are responsible for testing your work thoroughly prior to submission.
- Compiler warnings will result in a loss of points and should be remedied prior to submission of your final effort.
UNIX and Windows (or other operating systems)

1. We DO NOT encourage that you develop, compile, or to test your code in ANY editor or compiler other than those introduced in this course. Not all C compilers or versions of MATLAB/Octave are fully compatible.
   - You should want to test your work under the same conditions that it will be processed and tested by the course.
2. It is your responsibility to thoroughly test your code on the gcc compiler on guru.itap.purdue.edu as set up during the first week of the semester prior to submitting your work!
   - For MATLAB assignments test your work on guru.itap.purdue.edu with Octave, a program mostly compatible with MATLAB.
3. We will not support compilers, editors, and shells other than the default gcc compiler on ITaP’s guru machine, vi, and csh.
   - All students must be sure to set your host machine to: guru.itap.purdue.edu

Verifying a Successful Submission

How do you know if your assignment has been submitted as expected?

- Collaborative assignments such as labs are e-mailed to each member of the group based on the Purdue University e-mail addresses listed in the assignment header upon a successful submission.
  - Consider making a submission prior to leaving a lab session so that each of the group members has a copy of all work created during lab.
- Individual homework assignments are e-mailed only to the submitter upon a successful submission.
  - It is your responsibility to review the e-mail sent to verify you have submitted the correct file, to the correct lab section, and to the correct assignment.

You are advised to leave sufficient time prior to the assignment deadline to seek assistance should you encounter a difficulty with assignment submission. No late work will be accepted.