Teaching Portfolio

June 14, 2007

Jonathan Misurda

Department of Computer Science
University of Pittsburgh
6213 Sennott Square, 210 S. Bouquet St.
Pittsburgh, PA 15260

Voice: (412) 624-8835
Fax: (412) 624-5249
E-mail: jmisurda@cs.pitt.edu
Web: www.cs.pitt.edu/∼jmisurda

Contents

Teaching Philosophy

Computer Science as a discipline has a dichotomy in its focus. Half of the material is theoretical while the other half is practical application. Because this dichotomy can exist within even a single course, one of the most important jobs for any teacher of Computer Science is to bridge this gap. It is much too easy to lose a practically-oriented student while investigating the technical or mathematical background of a topic. Despite the seemingly huge disparity in the types of learning that these two aspects require, neither the theoretical nor the practical exist in a vacuum. They each drive the other, and it is most satisfying when as a teacher I can help students see the mutual influence, and thus gain a fuller understanding than just knowing either aspect alone.

As an instructor, I see my main purpose as being a resource to help solve problems and emphasize the application of the theoretical, while not losing sight of the higher-level concepts. Problem solving, both in the pedagogical sense, and in the sense of fixing the inevitable bugs associated with computer programming, is a skill that can benefit anyone. I see it as imperative that I not only show where problems may arise and how to prevent them, but also to teach students how to solve them on their own. There is no shortage of problems in Computer Science or in real life. Problem solving when applied to why a car will not start is no different than when applied to why a program will not run correctly. There is a logical process to investigate problems, and the students who internalize this process should succeed better in Computer Science and hopefully in life as well. It is my job to make this process evident and to instill its value and utility.

To ground and unite concepts, my lectures often start with a theoretical description of the new topic or problem to solve, followed by guided class participation attempting to construct a solution. With several solutions proposed, we then examine the tradeoffs that may make a certain approach better than another. These new concepts are put into practice through weekly labs, usually done in recitations. Quizzes, tests, and take-home projects review the learning progress made and serve to tie larger concepts together.

One of the main benefits of being a teacher while still a student is that I remember those aspects of teaching that have been most successful for me. My teaching reflects those teachers who have had the most influence upon my own education, and there seems no better way to reach my own students than to emulate their strengths and combine them with my own personal style.

As a teacher, I represent more than an opportunity to get a good grade. I strive to help all students gain a passion for Computer Science and learning in general, one that I myself have and attempt to demonstrate in both my teaching and my outside projects. There is a wealth of knowledge and understanding to be gained both about computers and life, much of which is to be gained outside of school. If I succeed in inspiring my students, they will learn on their own how to be better Computer Scientists.

Teaching Experience

Teaching Appointments

Instructor, Computer Science Department, University of Pittsburgh:

- CS 1550 – Intro. to Operating Systems, (Summer 2007)
- CS 0132 – Intro. to C and Guide to the UNIX System, (Fall 2005 and Fall 2006)
- CS 1590 – Social Implications of Computing Technology, (Summer 2006)
- CS 0004 – Intro. to Programming in Visual Basic, (Spring 2006)
- CS 0007 – Intro. to Computer Programming in Java, (Summer 2004 – Summer 2005)

Teaching Assistant, Computer Science Department, University of Pittsburgh:

- CS 1550 – Introduction to Operating Systems, (Fall 2004 – Spring 2004)
- CS 0401 – Introduction to Computer Programming in C++, (Summer 2001)

Other Teaching

Guest Lecturer:

- Developing for the NACHOS Educational Operating System, CS 1550, (July 1, 2004)

Tutoring:

- Clara Klein, CS 0004, Spring 2007 Term

Reflections on Teaching

During my experience as a graduate student, I have taught four different courses over eight semesters, and have also been a teaching assistant covering a recitation for two more semesters. Most of my experience as an instructor has been for service courses where I am introducing programming to students with little or no Computer Science background.

The majority of my teaching experience comes from two classes: CS 0007 – *Introduction to Java Programming* (4 sections) and CS 0132 – *Programming in C and a Guide to UNIX* (2 sections.) CS 0007 is geared towards freshmen who did not take any programming courses in high school. The class begins with a description of computers and how to write algorithms to solve problems. Take-home projects are usually casino games since they are interesting to write and test, and have very definite rules that are easy to express in a programming language. CS 0132 is taken by a variety of students who have had some programming experience before, but need to learn C. They are often adults or graduate students from other departments that need to learn C to do their work and research.

CS 0004 – *Introduction to Computer Programming in Visual Basic* was a very challenging course to teach. The majority of the students were either taking it to fulfill a graduation requirement for a Quantitative Reasoning class, or were in a technology-related but non-programming field such as Medical Information Management. Those students using it to fulfill a graduation requirement were taking the course in place of algebra, and were generally uncomfortable using math. The course greatly challenged these students, and I often provided extra time in office hours for those students who felt they needed additional help. I am proud of all of my students for successfully rising to the challenge of the final project, writing the game of Minesweeper. Programming Minesweeper required them to demonstrate the same abstract reasoning that they were trying to avoid by not taking algebra.

CS 1590 – *Social Implications of Computing Technology* was a wonderfully different experience to teach. The class consisted of majors who were generally upperclassmen, and the course's focus was on writing and thinking critically about the ethical issues of computers and the internet. Both the students and I had an interesting and insightful time learning and discussing controversial but important issues like net neutrality and copyrights.

Pedagogical Training and Contributions

Training

- FACDEV 2200 – University Teaching Practicum
- New TA Orientation

Contributions

New TA Orientation Presentations, University of Pittsburgh Arts & Sciences:

- Teaching Independently – (January 2, 2007)
- Dealing with Difficult Situations – (August 26, 2005 and August 26, 2006)
- Cheating and Plagiarism (Math, Natural Sciences, and Engineering) – (January 21, 2006, January 22, 2005, and September 11, 2004)
- Getting Started Non-international Teaching Assistants – (January 3, 2006 and August 27, 2004)
- Testing and Grading – (September 10, 2005)

Computer Science Department TA Orientation Presentations, University of Pittsburgh:

- 6 Things I Learned as an Undergraduate About Teaching – (September 2, 2005 and August 23, 2006)

Other Contributions, University of Pittsburgh:

- College in High School Materials for High School Introductory Java courses based upon my CS 0007 materials.

Evaluation

Awards

- 2006 *Orrin E. & Margaret M. Taulbee Award for Excellence in Computer Science.* Dept. of Computer Science, University of Pittsburgh. (Recognizes teaching excellence and academic progress toward a Ph.D.)
- 2005 *Orrin E. & Margaret M. Taulbee Award for Excellence in Computer Science.* Dept. of Computer Science, University of Pittsburgh. (Recognizes teaching excellence and academic progress toward a Ph.D.)
- Nominated to compete for the 2006 *Elizabeth Baranger Excellence in Teaching Award* (A student-nominated award for graduate student teaching.)

Student Opinion of Teaching Survey

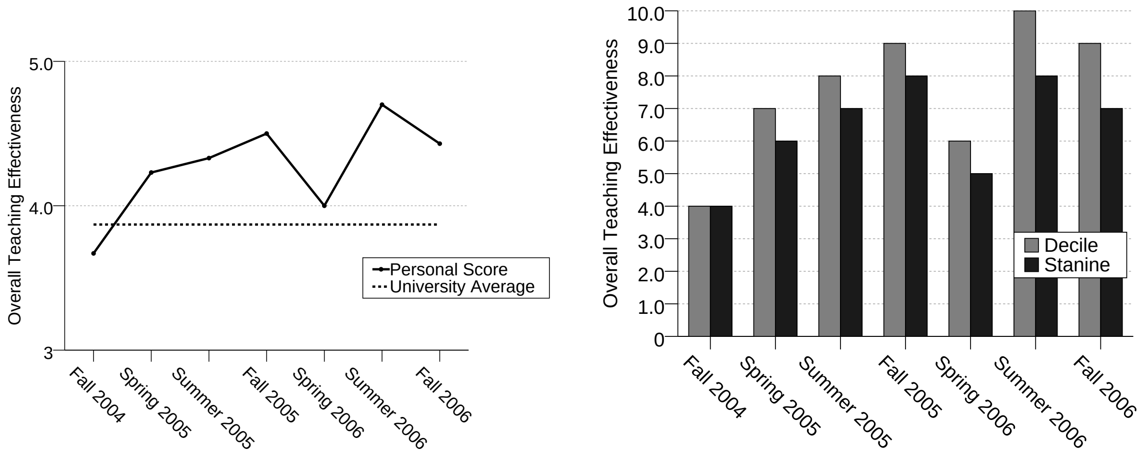

Figure 4.1: Overall Teaching Effectiveness evaluation scores over time.

Overall Teaching Effectiveness (Figure 4.1) is the student's chance to give a single overarching grade on an instructor's teaching that term. For the most part, my evaluated scores have been consistently improving and above the average. The slight dip that occurs in Spring 2006 is when I taught CS 0004 to a heavily non-major class for the first time. I still feel this was a good example of my teaching ability due to the general apprehension of the class toward anything resembling math. I look forward to being able to take what I learned from this course and apply it to similar ones in the future to see how I improve.

The right graph in Figure 4.1 shows how I compare statistically to a random sample of other University teachers. Decile breaks the scores into ten equally sized chunks based on score. The median would then be the end of the fifth decile. The stanine breaks the distribution of evaluations for the University into nine unequal chunks based upon a normal distribution. The frequency of high values for each of these measures indicates that I have consistently done better than the average teacher.

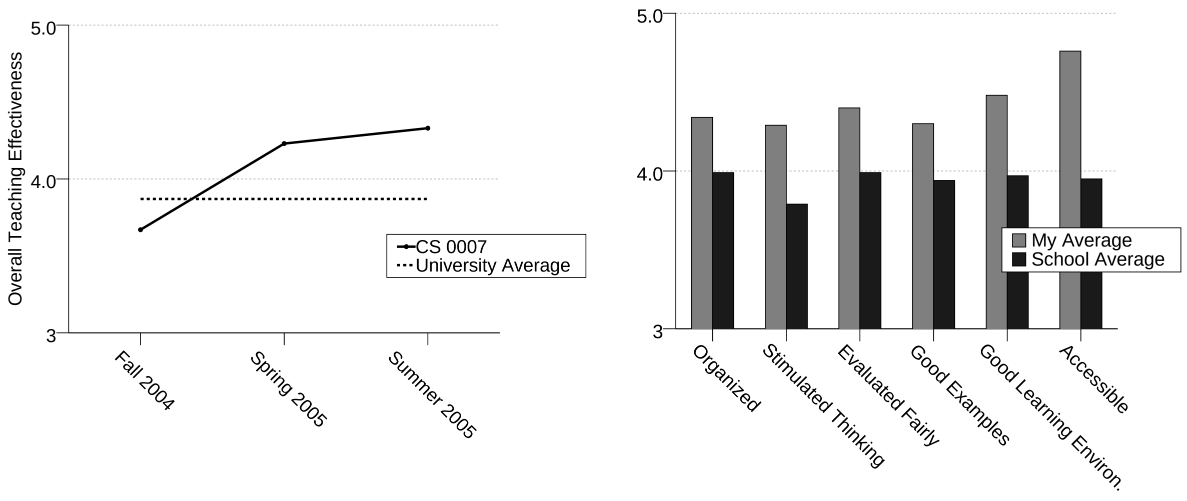

Figure 4.2: (*Left*) Improvement in CS 0007 over time. (*Right*) Personal averages for other student-evaluated criteria

Not only do my evaluations generally increase as I have gained more experience, but my evaluations for several sections of the same course also show consistent improvement (Figure 4.2(Left).)

In addition to overall teaching effectiveness, students are also asked to respond to several other specific aspects of teaching. Figure 4.2(Right) shows these additional questions and my personal averages. I consistently rank in the upper deciles for these values. One particular standout is my accessibility, which reflects my willingness to meet students in office hours, and respond to their questions via email in a timely fashion.

The final question of the student evaluation of teaching is whether the student would recommend the instructor to other students. Over all of the classes I have taught, 95% of students who responded to the question would recommend me as an instructor, including over 65% who would definitely recommend me to other students.

Student Comments

Sample responses to "What were the instructor's major strengths?"

- Prof. Misurda is one of the best teachers at this University, His class is always taught in a well-organized, effective manner. The course material he chooses has proved to be extremely helpful in other CS classes. He is always accessible and helpful to his students. In short, this University needs more teachers like Prof. Misurda. —*CS 0132 Student, Fall 2006*
- Tried to teach the course on the level of the students and explained abstract/logical ideas very well; I learned much from this class even after taking a year and a half of computer programming in high school. —*CS 0007 Student, Spring 2005*
- Jonathan is an excellent instructor. His lectures make very difficult topics crystal clear and easy to remember and use. I wish I could take another one of his classes because I know I would learn a lot. He tailors the course, and lectures, to fit the needs of the class. —*CS 0007 Student, Spring 2005*
- Very friendly and knowledgeable. Actually concerned with his students learning and understanding the material, which is rare. Very down to earth. Made lectures interesting and enjoyable (and humorous!) —*CS 0132 Student, Fall 2006*
- He was very patient with everyone, especially with people who were having a hard time with some of the concepts. —*CS 0007 Student, Summer 2005*
- Extremely accessible and helpful. Enthusiasm in class. Genuine interest in student success. —*CS 0004 Student, Spring 2006*

Example Teaching Materials

The actual documents referred to here can be found in the indicated Appendix at the end of this teaching portfolio.

Syllabus – Appendix A

In all of the pedagogical training I have received, there has been a significant emphasis on the importance of a well-made syllabus. It serves not only to inform the student about what requirements and scheduled topics an instructor forsees for the term but also forms an agreement that protects students and teachers alike should a dispute arise.

The first time teaching a course is often the most difficult time to create a good syllabus. The one I selected here represents the first and only time I taught this particular course, and the overly optimistic schedule of topics would certainly be altered if I have the opportunity to teach this course again. I would also further elaborate on what sort of time expectations the associated out-of-class work carries. This was a point that many students felt anxious about, and something I will commit to paper in the future.

I did, however, emphasize the technological requirements of taking this course. Doing these programs at home required a significant software installation, even though the program itself was available for free to the students. Also, due to the data storage requirements, a USB drive was required, as opposed to something cheaper but much more fragile such as floppy disks. Since this was a small but unexpected cost to the students, I wanted to make sure that this was emphasized early.

Project – Appendix B

Take-home projects represent the bulk of the experience in a programming course. Keeping these projects interesting is a challenge for any teacher of Computer Science. I have learned one good way to encourage students enrolled in a second or later programming course to practice their programming skills is by competing in games. The project that I show here is the game of Mastermind, where one player creates a solution that the opponent must guess given only imprecise feedback.

This type of project emphasizes functions and good software design, because each student will be providing their solution as an encapsulated part of the program rather than having control over the entire project. This results in a need to define a consistent interface, and teaches conditional compilation and Makefiles.

Students enjoy the project immensely, and I am always excited to see the creative solutions that they come up with. One example of a great solution was by a student who realized that the solution was stored in a stack allocated variable, and thus was able to "cheat" by direct manipulation of pointers. This made me as an instructor realize the success that I had in explaining how local variables' lifetime and scope worked.

Final Exam – Appendix C

This represents my first attempt to create a final exam. I drew upon two separate but complimentary sets of guidelines to govern question content and types. The first was Bloom's Taxonomy [1] which attempts to classify the different levels of learning and understanding to allow educators to strive for a more complete treatment of the topic. The other source of guidance was the experiences my friends and I had taking tests throughout our schooling. One of the major points of agreement was that I should vary the question type and not focus on any one style of question more than another. This helps cover different levels of learning as well as to not penalize any student who has difficulty with a particular type of question. The five major types of questions (multiple choice, short answer, code tracing, code writing, and short essay) that I use here have become a framework for subsequent tests that I truly feel works well and provides feedback on student learning that is a good indicator of their understanding.

This test was the third of three exams in the semester, a number that I felt allowed no one test to be too significantly weighted in the class. I found, however, that the first test did not cover enough material for me to find it particularly useful. From this point on in my teaching, I switched to two tests per term, but increased the value of the take-home projects to avoid having tests be worth too much and cause test anxiety.

[1] *Taxonomy of Educational Objectives: The Classification of Educational Goals*; pp. 201-207; B. S. Bloom (Ed.); Susan Fauer Company, Inc. 1956

Appendix A: Syllabus, CS 0004 – Spring 2006

CS 0004 - Introduction to Programming with Visual Basic

Spring Term: 2064

	Class
Time:	4:00 – 5:15
Days:	TH
Room:	5505 Sennott Square
Webpage:	http://www.cs.pitt.edu/~jmisurda/teaching/cs0004.htm

Contact Information

Instructor: Jonathan Misurda	
Office:	6404 Sennot Square
Phone:	(412) 624-9129
Email:	jmisurda@cs.pitt.edu
Office Hours:	TBA

Description

Computers have become increasingly pervasive in today's society. Their power to do complicated and repetitive tasks quickly and efficiently has made them invaluable tools in modern society. In this class, you will learn to harness the power of a computer to do new tasks by creating your own software as opposed to using existing programs. We will explore the limits of what a computer can and cannot do easily, and provide you with the knowledge and experience to recognize those problems that you may find in life that can be solved with the help of a computer, and the ability to make the computer do those tasks.

This class is meant as a first class in computer science. Anyone is welcome to take it, as computers can be useful in a variety of fields. It is also meant as an introduction to computer science for those students who wish to pursue the field further, but lack prior exposure to the material.

Prerequisites

There are no prerequisites. Concepts from basic Algebra will be drawn upon. Passing familiarity with the operation of a computer is also helpful.

(If you are already proficient – i.e. had 1 or 2 courses prior – in computer programming this is not the course for you.)

Disability Resources and Services:

If you have a disability for which you are requesting an accommodation, you are encouraged to contact both your instructor and Disability Resources and Services, 216 William Pitt Union, (412) 648-7890, as early as possible in the term. DRS will verify your disability and determine reasonable accommodations for this course.

Course Purposes and Goals

Introduction to Computer Programming is meant as a course to expose interested students how computer software is written and tested. It seeks to illuminate the creative process that is computer programming from the ground up, with an emphasis on good preplanning and style.

In this course we will be writing event driven programs for the Windows Operating System in the Visual Basic (VB) programming language. Goals for the course include:

- Being able to write computer programs in VB that can perform a variety of tasks.
- Recognizing and creating well-written programs in good programming style
- Participating in the various stages of the life-cycle of a computer program, including planning, implementation, and debugging

Textbook

The text can be found in the bookstore:

Schneider, David I. *An Introduction to Programming Using Visual Basic .NET* – Fifth Edition. Prentice Hall, New Jersey, 2003. ISBN: 0-13-030657-6

Additional Materials

In addition to the textbook, you will also need some type of removable storage media to store your programs on while you work on them. I suggest a small, inexpensive USB thumb drive. Anything over 16 MB should be sufficient for the work in this course. Floppy disks or rewriteable CDs are also useable, but are a more brittle form of storage meaning you may lose your work during the term.

Since this is a computer class with out of class assignments, you will also need access to a computer that has Visual Basic .NET, Visual Basic .NET 2003, or Visual Basic 2005 installed on it. All campus lab computers are already installed with Visual Basic .NET 2003. If you want to work at home, we can discuss ways of doing this. Visual Studio .NET 2003 is available on CDs for free from the campus labs, Visual Studio .NET can be downloaded from http://software.pitt.edu (however it is huge), and Visual Basic 2005 Express can be downloaded for free from Microsoft.

Class Policies

Exams: There will be a midterm and a final in this class. The final exam is scheduled for Monday, April 24th 2006, from 2:00 p.m. – 3:50 p.m.

Cheating on exams will not be tolerated. Anyone caught cheating will be given a zero for the test and reported to the department following University procedures.

Quizzes: There will be unannounced quizzes given throughout the term.

Projects: There will be four out-of-class assignments given. These are to be completed in the given time (no extensions will be given without a valid excuse.) These are meant to be your own work; anyone found to be collaborating will be given a zero for the assignment. Collaborating also means using code from previous terms, other universities, your friends, or finding it on the internet.

Participation: Attendance will not be taken, but in a small class, any absence will be noticed. Several unexcused missed classes will adversely affect your grade. Asking and answering questions will also be considered as part of the participation grade.

Grading

Your grade will be based upon 2 exams, 4 projects, quizzes (the lowest one will be dropped), and participation:

2 Exams	40% (20% each)
4 Projects	40% (10% each)
Quizzes	10%
Participation	10%
Total	100%

The scale for the term will be:

Percentage	**Letter**
97 or above	A+
93-96	A
90-92	A-
87-89	B+
83-86	B
80-82	B-
77-79	C+
73-76	C
70-72	C-
67-69	D+
63-66	D
60-62	D-
less than 60	F

Term Schedule: The daily topics are subject to change depending on our pace. They are there to assist you in the readings so you can focus on those concepts prior to class.

Week 1 – (1/5)

Readings for this week: Purchase the book

Topics:
- Introduction to the Course
- Demonstration of Visual Basic

Week 2 – (1/10)

Readings for this week: Chapter 1

Topics:
- Event driven programming
- Windows Basics
- Examination of dialog-based and view-based applications
- Common Windows controls
- Interface design in VB

Week 3 – (1/17)

Readings for this week: Chapter 2

Topics:
- Problem Solving
- Algorithms
- Flowcharts

Week 4 – (1/24)

Readings for this week: Chapter 3, Appendix D

Topics:
- Controls and Events under VB
- Stepping through a program
- Numbers and arithmetic, Strings

Week 5 – (1/31)

Readings for this week: Chapter 4

Topics:
- Reading the online documentation
- Subroutines
- Functions
- Modules

Week 6 – (2/7)

Readings for this week: Chapter 5

Topics:

- Linear control flow
- If - Then Blocks
- Select Case Blocks

Week 7 – (2/14)

Readings for this week: Chapter 6

Topics:

- Repetition
- Looping (Do…while, For…next)

Week 8 – (2/21)

Readings for this week: Chapters 1-6

Midterm Exam – February 23rd, 2006 during class

Topics:

- Review for exam

Week 9 – (2/28)

Readings for this week: Chapter 7

Topics:

- Arrays
- Sorting and Searching
- Multidimensional arrays

SPRING BREAK – (3/7)

No Classes This Week

Week 10 – (3/14)

Readings for this week: Chapter 8

Topics:

- Files
- Text and Binary files
- Sequential vs. Random Access

Week 11 – (3/21)

Readings for this week: Chapter 9

Topics:
- Additional Windows Controls
- List Boxes
- Common Controls

Week 12 – (3/28)

Readings for this week: MSDN/Online Help

Topics:
- Interfacing with the Win32 API
- ShellNotifyIconData

Week 13 –(4/4)

Readings for this week: TBA

Topics:
- Good programming practices
- Examples of do’s and don’ts

Week 14 – (4/11)

Readings for this week: TBA

Topics:
- Visual Basic for Applications
- Office Macros

Week 15 – (4/18)

Readings for this week: *None*

Topics:
- Review for the Final Exam

Finals Week

Final Exam – Monday, April 24th 2006, from 2:00 p.m.- 3:50 p.m. in 5505 Sennott Square (Normal classroom)

Appendix B: Take-home project, CS 0132 – Fall 2006

CS 132 – Project 3: Mastermind

Due: Monday, November 6th by the start of class

Introduction

In this game, your opponent chooses 4 pegs each with one of 6 colors. Your job is then to guess the colors that your opponent has chosen in the proper order. After each guess by you, the computer will give two numbers as feedback. The first number is how many pegs are the proper color and in the proper position. The second number is how many pegs are the proper color, but not in the correct position. Once you get the pattern correct, you and your opponent switch tasks, with you providing the solution, and them guessing. Whoever guesses the solution in the fewest guesses wins.

Project Description

This project is a friendly competition. Your job is not to write the entire game of "Mastermind," as I've done that for you. Your job is to come up with the intelligence behind the players. You will write two functions: one that creates a solution and one that takes a guess. I will take all of your submissions and combine them with my driver program, and pit you in a head-to-head battle against the code written by your classmates. The winner will receive a prize.

Design-by-contract

Since you will be combining your code with code you did not write, we must agree upon some common interface so that the parts will work when we put them together. We will use C's facilities for multi-file development, as well as the preprocessor directives to achieve this.

This agreement on how to interface the components together is referred to as Design-by-contract in Software Engineering.

The contract for this particular project is as follows:

```
#ifdef CHOOSER
void create_solution(int solution[]){
      return jrmst106_solution(solution);
}
#endif

#ifdef PLAYER
void create_guess(int right, int almost, int guess[]) {
      return jrmst106_turn(right, almost, guess);

}
#endif

void jrmst106_solution(int solution[])
{
      solution[0] = RED; // a silly solution
      solution[1] = RED;
      solution[2] = RED;
      solution[3] = RED;
}

void jrmst106_turn(int right, int almost, int guess[])
{
      guess[0] = rand()%6; //random opponent
      guess[1] = rand()%6;
      guess[2] = rand()%6;
      guess[3] = rand()%6;
}
```

You must make a file named your_username.c The file shown above is jrmst106.c (all lowercase please.) Inside of it you must implement two functions: `create_solution` and `create_guess`. When you submit your program, each of these functions should simply call a function that does the actual work, but is prefixed with your username. In this example, my program simply picks random guess and creates a fixed solution. You will play against your classmates 3 times as the chooser and 3 times as the guesser.

The driver program (which we will dissect in class) will then play the two opponents against each other until a winner is determined.

Development

In order for you to test your function, I am providing you with all of the driver code. Under your `private/cs132` directory, make a directory called `project3` and `cd` into it. There execute the following:

```
cp  ~jrmst106/public/cs132/mastermind.tar  .
tar  xvf  mastermind.tar
```

This will create 3 files: `Makefile`, `mastermind.c`, and `jrmst106.c`

Type `make` and then run the `mastermind` program to see it in action.

You can then make a copy of `jrmst106.c` with your username as the filename. Make sure you update the `Makefile` to compile the new file.

Hints

- Play the game with a human opponent and see if you can develop a good strategy
- Since you are making your own file, you can make file-scoped variables to keep track of information between calls of your function if you would like
- In testing your function, you can make one of the functions just read input from the keyboard.
- Your code can try to cheat, however my code prevents some obvious ways of doing so. I reserve the right to modify the driver at anytime to stop an avenue for cheating that I see, up until the time of submission.
- That said, it's more likely you'd benefit from a good player strategy rather than devoting a ton of effort to find a cheat.
- `create_guess` gets the almost and right values from the previous guess. You probably want to remember that guess so you can use the information to help construct your new guess
- The longest the game should ever take is 24 guesses, however it has been proven that no game should take beyond 7 guesses by a truly intelligent player. Try your best!

Requirements

By the deadline you must submit a tarfile containing:

- A file named `username.c` that contains an implementation of the `create_solution` and `create_guess` functions in the manner of the example above
- A `Makefile` that builds your program

Appendix C: Final Exam, CS 0007 – Summer 2004

Name: ________________________

CS 0007 – Final Exam

Directions: You have 1 hour to complete this test. Remember to keep your eyes on your own paper. Make sure everything is off your desk. Good luck!

Multiple Choice (15 points)

1.) Which of the following is a legal identifier in Java?

A) %off

B) 4ever

C) total_count

D) String

2.) Which of the following is the keyword that stops a loop from repeating and skips the remaining statements in the loop?

A) goto

B) break

C) continue

D) skip

3.) In a switch statement, what is used in the same manner as "else" in an if-else if-else statement?

A) default

B) catch

C) case else

D) select

4.) What is the appropriate function to convert a String to an int?

A) String.parseInt()

B) java.util.parseInt()

C) Integer.parseString()

D) Integer.parseInt()

5.) Which of the following will return the length of array a[]?

A) a.size()

B) a.last()

C) a.length

D) a.max

Short Answer (25 points)

6.) Write code to produce a random number from -10 to 10. (5 points)

7.) What is the value of the following expressions? (5 points)

A) 3 * 5 + 4 * 6 % 2

B) “Hello ” + 3 + “ there”

C) What are i and j after the following executes?

```
int i = 5;
int j = i++;
```

8.) Finish the factorial function using recursion: (10 points)

```
int factorial(int n) {









}
```

9.) Put the following measures of algorithmic complexity in order from least to greatest (use < to say A < B < …): (5 points)

O(N), O(log N), O(1), O(N^2)

This space intentionally left blank. Go on to the next page.

Tracing (20 points)

10.) What is printed on the screen from the following program? (10 points)

```
class Test {
        public static void main(String args[]) {
                int a[] = {3, 4, 7, 5, 8, 9, 2, 5};
                int q = 0;


                for(int i =0; i<8;i++) {
                        q += a[i];
                }
                q /= 8;

                System.out.println(“The answer is ” + q);
        }
}
```

11.) What is the output of the following program? (10 points)

```
class Mystery2 {

     public static void main(String args[]) {
          int a[] = { 3 , 6 ,7 ,2 ,3 ,4,  1 ,8 , 10};

          s(a);

          for(int i=0; i<9;i++) {
               System.out.println(a[i]);
          }

     }

     public static void s(int []b) {
          for(int i=0;i<9;i++) {
               int temp = m(b, i);
               t(b, i, temp);
          }
     }

     public static void t(int []c, int l, int r) {
          int temp = c[l];
          c[l] = c[r];
          c[r] = temp;
     }

     public static int m(int d[], int l) {
          int temp = l;
          for(int i=l;i<9;i++) {
               if(d[i] < d[temp]) {
                    temp = i;
               }
          }
          return temp;
     }
}
```

Coding (30 points)

12.) Write a program that takes a word on the command line, splits it into characters (hint: use toCharArray()) and prints it out letter by letter in reverse. (10 points)

13.) Write a program that fills a two dimensional array with the following diagram and then prints the array to a file. Make sure it works for any defined SIZE. I have started it for you: (20 points)

Your goal:

```
    .
   . .
  . . .
 . . . .
. . . . .
```

```
import  java.io.*;

class EveryonesFavoriteTriangle {

          private static final int SIZE = 5;

          public static void main(String args[]) throws IOException {
                    PrintWriter out = new PrintWriter(new FileWriter("output.txt"));
```

Essay (20 points)

14.) What does it mean to say a program is Event-Driven? How does this compare to a console application? (10 points)

15.) What is encryption? Why is it useful? Why are computers particularly good at doing it? (10 points)