University of South Carolina
Carolina Core

**COMPUTER SCIENCE 145**
**ALGORITHMIC DESIGN I**

**BULLETIN INFORMATION**
CSCE 145: Algorithmic Design I (4 credit hours)
**Course Description:**
Problem-solving, algorithmic design, and programming. Open to all majors.
Prerequisites: Placement in MATH 141 or grade of C or better in MATH 115
**Note:** Three lectures and two laboratory hours per week. Open to all majors.

**SAMPLE COURSE OVERVIEW**
This is the first course in Computer Science and Engineering. It introduces the design of computer algorithms and their implementation in the Java programming language.

**ITEMIZED LEARNING OUTCOMES**
**Upon successful completion of Computer Science 145, students will be able to:**

1. Solve problems using a computer
2. Read and design algorithms
3. Design data structures
4. Demonstrate the ability to use a software development environment to construct, execute, test, and debug software
5. Demonstrate the ability to program a computer in a high-level language

**SAMPLE REQUIRED TEXTS/SUGGESTED READINGS/MATERIALS**

1. Walter Savitch and Frank Carrano, Java: Introduction to Problem Solving and Programming (5th Edition), Prentice Hall, 2009, ISBN-10: 0136072259.
   Students use the 'eclipse' IDE which is installed in the computers in the lab.

**SAMPLE ASSIGNMENTS AND/OR EXAMS**

1. 2 weekly laboratories (about 25 labs in total). These are short programming projects to be done in the computer lab.
2. Weekly or bi-weekly homework assignments (about 10 homeworks in total). These are longer programming projects to be done by the students at home, working either alone or in pairs.
3. 2 or 3 in-class tests and a final test. These are open-book tests to be done individually during class time.
4. The topics of the homework and labs mirror the Topics Covered above.

    a. **Sample Homework 6:**

For this problem you will implement a program that asks the user to type in a random sequence of 0's and 1's and then prints a distribution of all the length-3 substrings in his input, as well as the standard deviation in these numbers. The following is a sample interaction of the user with the program:

Enter random sequence of 0 and 1s:
00000111
The distribution of length=3 substrings is:
000 3
001 1
010 0
011 1
100 0
101 0
110 0
111 1
Deviation = 0.9682458365518543

**b. Sample homework 2:**

For this homework you will implement a variation of the Mastermind board game, where we are using digits instead of colors and changing the rules a little bit. In this game your program will generate a random 4-digit number that the user then has to guess (use Math.random() to generate a random number). After the user makes a guess your program will tell him:

how many of the digits in his guess are correct, that is, they have the same value and in the same position as in the secret,
how many digits in his guess are not found in the secret.

Below is a sample run. Notice that the program is telling the user the secret. This is useful for debugging. The program you turn in should also tell the user the secret so as to make it easier for us to grade it.

**Guess the 4-digit number I am thinking of.**
**(the secret is 9712)**
**Your guess:1245**
**1245 has 0 digits in the correct position.**
**1245 has 2 digits that are not found anywhere in the secret number.**
**Your guess:9999**
**9999 has 1 digits in the correct position.**
**9999 has 0 digits that are not found anywhere in the secret number.**
**Your guess:5555**
**5555 has 0 digits in the correct position.**
**5555 has 4 digits that are not found anywhere in the secret number.**

**Your guess:9779**
**9779 has 2 digits in the correct position.**
**9779 has 0 digits that are not found anywhere in the secret number.**
**Your guess:9721**
**9721 has 2 digits in the correct position.**
**9721 has 0 digits that are not found anywhere in the secret number.**
**Your guess:1234**
**1234 has 0 digits in the correct position.**
**1234 has 2 digits that are not found anywhere in the secret number.**
**Your guess:9712**
**9712 has 4 digits in the correct position.**
**9712 has 0 digits that are not found anywhere in the secret number.**
**Congratulations! You guessed correctly.**

**c. Sample Lab 19:**

For this lab you will implement a recursive method that draws the Sierpinsky Triangle to various levels.

**d. Sample Lab 12:**

For this lab you will write a program that first generates a random cypher key and then uses that key to encrypt the users' messages. The cypher key is just a mapping from every letter in English (just the lower case letters) to another one.

A simple way to generate a random cypher is the following
Create a char[] cypher array where cypher[0] = 'a', cypher[1] = 'b' and so on. Remember that if i is an integer then (char)('a' + i) will be 'b' when i==1, 'c' when i==2, and so on.
Then, to randomize cypher simply pick 2 random indexes in the array and swap the values. Repeat this 100 times and the array should be properly randomized.

You will then use this cypher array to encrypt the user's message. Below is a sample interaction with the user:

**Our cypher is:**
**abcdefghijklmnopqrstuvwxyz**
**jbeqkmrvayihonpfgdsxutczlw**
**Enter your message:**
**abc xyz**
**The encrypted message is:**
**jbe zlw**
**Enter your message:**
**Launch code Alpha, Tango, Tango, Bravo, Macho**

**The encrypted message is:**
**hjunev epqk jhfvj, xjnrp, xjnrp, bdjtp, ojevp**
**Enter your message:**
**how about a nice game of chess?**
**The encrypted message is:**
**vpc jbpux j naek rjok pm evkss?**
**Enter your message:**

Tip: Remember to toLowerCase his input and ignore spaces and other characters. You will find that either charAt or toCharArray will come in handy

Tip: If you have char c; then cypher[(int)('a' - c)] will access the 0th position of cypher when c=='a'.

**SAMPLE COURSE OUTLINE WITH TIMELINE OF TOPICS, READINGS/ASSIGNMENTS, EXAMS/PROJECTS**
TBA