**ECE 4318 Final**

Name: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date:** 12/12/19, Thursday, 3 p.m. – 4:50 p.m., at 163-2005

**Total Points**: 114 – 136 points

**Total Grade Points:** 20 grade points. 100 points = 20 grade points; points above 100 points are extra credits.

**Note:** you may redo or do up to 3 **(three)** questions of this final exam and submit by Friday 12/13/19, 5 p.m. Clearly specify which 3 questions you want to be reconsidered. Resubmission without specifying, without showing clearly your intention or with more than 3 questions resubmitted will not be considered. This is software engineering exam, so you need to turn in code and also type the answers.

Note: You are asked to write code in C in several questions. However, you are allowed to use some other programming languages that you are more comfortable such as Python. The same for

**Structured Programming cases**

*Matrix processing example* by Nyhoff (15%)

1. (15%) Code Analysis and reengineering. Please do this one in C programming language. When doing in class, you are **not** required to compile and run code with output, but you are required to turn in your reengineered source code

We have this piece of *spaghetti* code as an exercise of Nyhoff’s data structure book (ECE 304) (chapter 1).

**int row = 0, col;**

**A: col = 0;**

If (col < n) goto B;

goto A;

B: if (row < n) goto C;

goto E;

C: if (mat[row][col] == item) goto D;

col ++;

If (col < n) goto B;

row ++;

goto A;

D: cout << “item found\n”;

goto F;

E: cout << “item not found\n”;

F: ;

1. **(3%) Follow** the logic of this program and explain what it intends to do (does this compute a row sum, do a linear search, a binary search, a sort etc.?). Call this code C1. This is spaghetti code.
2. **(6%) Reengineer** or convert this to *structured* code. Call this code C2. **Turn** in this code as Q1.c. **Explain** the difference of structured code from the unstructured or spaghetti code.
3. **(6%)** Use a 5x5 matrix M you provide to explain how the code works. All entries of M must be nonzero

**Linux driver spaghetti code from StackOverflow.com web site (12%)**

Below is part of a Linux SCSI driver code (called C3) with line numbers (incomplete). Full code is LinuxDriverExample.docx file in the distribution folder.

1: wait\_nomsg:

2: if ((inb(tmport) & 0x04) != 0) {

3: goto wait\_nomsg;

4: }

5: outb(1, 0x80);

6: udelay(100);

7: for (n = 0; n < 0x30000; n++) {

8: if ((inb(tmport) & 0x80) != 0) { /\* bsy ? \*/

9: goto wait\_io;

10: }

11: }

12: goto TCM\_SYNC;

13: wait\_io:

14: for (n = 0; n < 0x30000; n++) {

15: if ((inb(tmport) & 0x81) == 0x0081) {

16: goto wait\_io1;

17: }

18: }

19: goto TCM\_SYNC;

20: wait\_io1:

21: inb(0x80);

22: val |= 0x8003; /\* io,cd,db7 \*/

23: outw(val, tmport);

24: inb(0x80);

25: val &= 0x00bf; /\* no sel \*/

26: outw(val, tmport);

27: outb(2, 0x80);

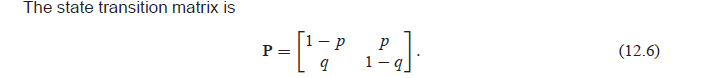
28: TCM\_SYNC:

1. (12 - 24%) **Convert** this Linux driver partial code to structured code Q2.c
2. (4%) Lines 1 through 4 is a loop using goto. Convert this to a normal while or for loop. Show the code.
3. (8%) If we comment out line 12, then lines 7 through 13 will look like a for loop with a break. Convert this modified code snippet of lines to structured code.
4. (12%\*) **Try to convert** the whole code of lines 1 through 28 (line 12 unmodified) to structured code. Do as much as you can. How many loops and conditional constructs (if else) do you find?

**Matrix processing (software) of Markov chain concept in ECE 3715 (probability and statistics). (10%)**

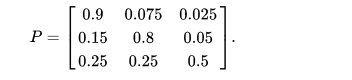
Markov chain is a well known math concept for engineers (especially ECE). In matrix form, a Markov chain M of dimension 2 is a 2x2 matrix with all the entries nonnegative and the row sums equal to 1. In a different book or literature, a Markov chain N of dimension 2 can be a 2x2 matrix with all the entries nonnegative, but the column sums equal to 1, but row sums may not equal to 1. In other words, a Markov chain N in this sense is equal to the transpose of Markov chain M mentioned earlier.

An example of 2x2 Markov chain with row sum = 1 comes as below from Yates and Goodman’s probability book as follows (2nd edition, page 449)



An example of a 3x3 Markov chain with row sums = 1 (and column sums may not equal 1) looks as below from Wikipedia:

<https://en.wikipedia.org/wiki/Markov_chain>



Note some other books or literature use the transpose of matrix P (called P’ or P t in some books) for 2x2 matrix and 3x3 matrix as Markov chain. In that case, the Markov chain has column sums equal to 1, but row sums may not equal 1.

1. (15%) **Write** a C program Q3.c so that
2. (4%) It **verifies** that the 2x2 matrix P in (12.6) and the 3x3 matrix P from Wikipedia are Markov chain.
3. (5%) **Give** an example of a 5x5 matrix P satisfying Markov condition
4. (6%) **Enhance** the program form part (a) so that it can tell if this is a Markov chain with row sum = 1 (and column sum may equal 1 or not) or a Markov chain with column sum = 1 (and row sum may equal 1 or not).
5. (12 - 22%) The following is the so called E 23(1,1) in cryptography or the set of all integer solutions of y2 = x3 + x + 1 (mod 23). as in the table below (there are 27 points of this curve)

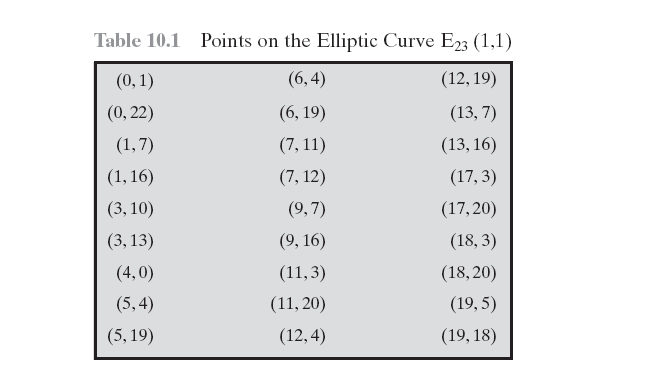


Figure 1

1. (12%) Write a program Q4.c, Q4.cs, Q4.Java, etc. to generate the 27 solutions of the equation y2 = x3 + x + 1 (mod 23) programmatically. The output format does not have to be the same as in Figure 1 above.
2. (10%) Generalize your program so that it can compute and display solutions of y2 = x3 + x + 1 (mod p) programmatically for p = 23, 29, and 31.
3. (23%) Object Oriented Programming class Complex (C# / Java / Python etc.). Turn in Q5.cs, Q5.java etc.
4. (2%) **Write** a program that defines a class Complex with the following methods, add () to add two complex numbers, mul () to multiply two complex numbers, and print () to print a complex number.

To help you start, a C# solution with code of add, mul, and print (and test) is provided as ComplexConsole.zip. You can use this or you can write your own C# code, your own C++, Java, Python etc.

1. (2%) **Enhance** the code in part (a) to have sub () to subtract a complex number from another one, div () to divide a complex number by another one (make sure the divisor is not zero)
2. (6%) **Enhance** by adding a method *polar* to convert the complex number a + bj to the polar form rei . Also add methods to calculate the absolute value or **magnitude** of a complex number a + bj, which is ; and also a method to calculate the **argument** or angle of a + bj, which is arctan (b/a).
3. (2%) **Use** your program to compute (1 + j ) / (1 – j), where j is
4. (6%) **Use** your program to compute (1 + j) 100 and  300 where  = (so that 3 = 1).
5. (7%) Consider the digital low pass filter used in ECE 306 / ECE 3101, H(z)= , with a = 0.9, and z changes through unit circle z = ei. (so H(z) = = . **Use** a = 0.9 and seven values of angle , calculate the magnitudes and angles of H(z) at andwith the two methods **magnitude** and **argument** you code in part (c) above. **Tabulate** the answers.
6. (16%) Test cases (do you know what test cases mean?)
7. (4%) **Provide** test cases for n!; for Fibonacci numbers
8. (3%) **Provide** two more test cases for one of Q1 **and show how those work** (with outputs of your code)
9. (3%) **Provide** two more test cases for one of Q3 and show how those work (with outputs of your code)
10. (3%) **Provide** two more test cases for one of Q4 and show how those work (with outputs of your code)
11. (3%) **Provide** two more test cases for one of Q5 and show how those work (with outputs of your code)
12. (21%) Software evolutions and software pricing
13. (5%) **Explain** the concepts of software evolutions, especially legacy systems and software maintenance
14. (4%) **Give** two examples of legacy systems and **explain**.
15. (4%) **Give** two examples of software maintenance and **explain**.
16. (4%) **Explain** the concepts of software pricing and also its significance
17. (4%) **Give** two examples of software pricing and **explain**.