**ECE 4318 Final**

**Date:** 5/15/21 Saturday

**Due Date:** 5/19/21 Wednesday 2 p.m. Please let me know if you need extension of 1 to 2 days

**Total Points: 153** points

**Total Grade Points:** 20 grade points, 100 points = 10 grade points

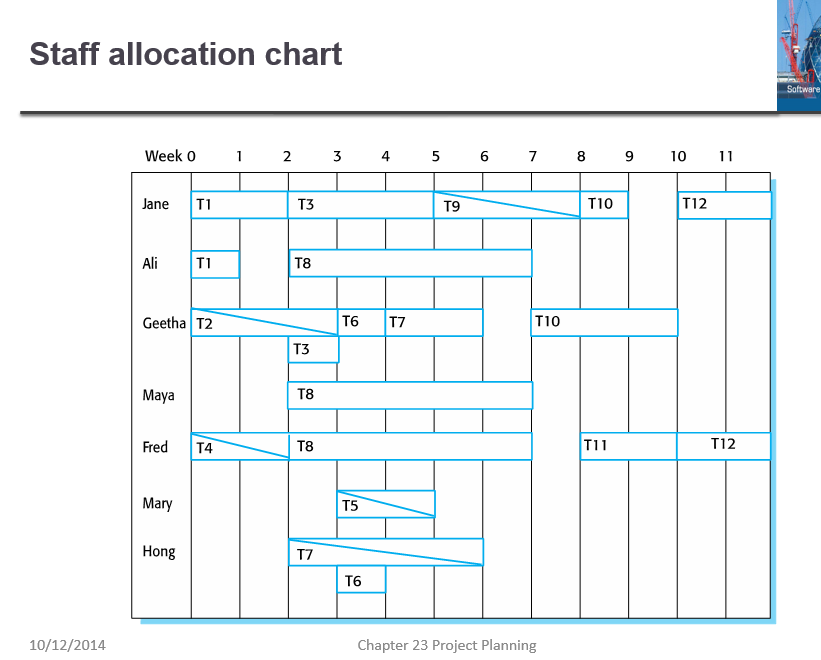
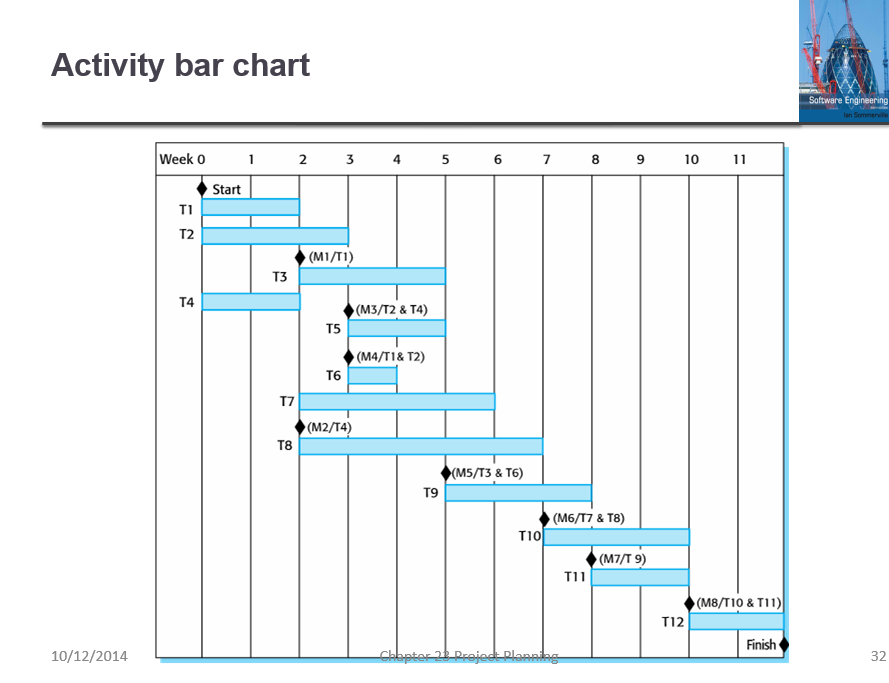
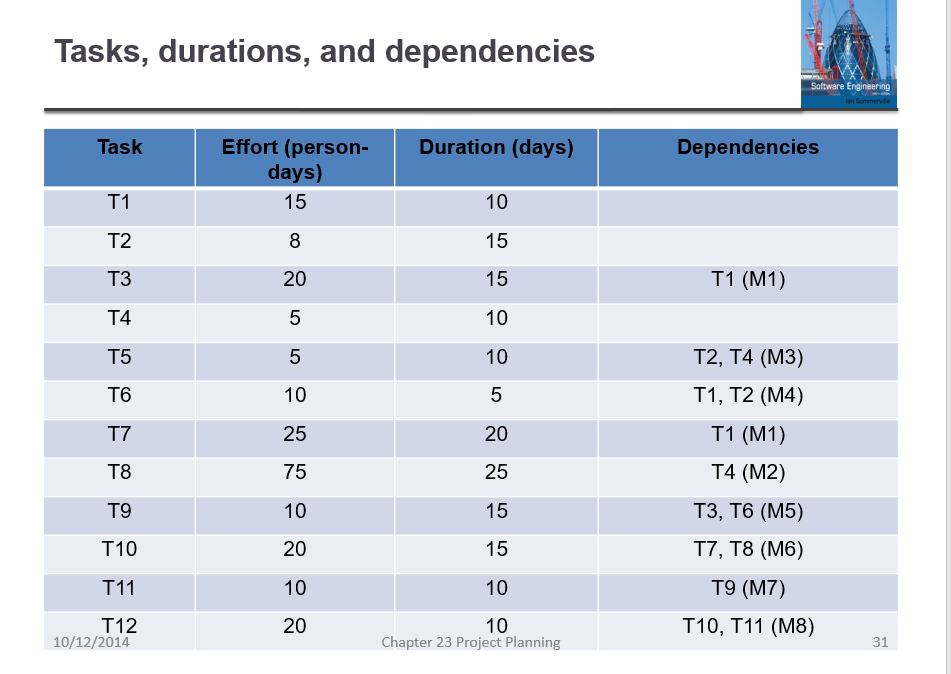
You will work alone (take home).

**Submit to Blackboard**

**Scheduling (43%)**

1. **(43% +) Scheduling (in Projects)**

Recently you have learned scheduling (a project) from chapter 23 Project Planning with slides 31, 32, and 33 as follows:



Slide #31 #32 #33 of chapter 23, Sommerfeld

**Enlarge** these slides if necessary (I reduced them here) for you to read.

1. **(20%) Verify** if these 3 slides present consistent data. There is possibility that these 3 slides are not consistent. For example, it is possible that T1 shows efforts of 15 person-days and duration of 10 days in slide 31, but in slide 32 Gantt charts it shows longer or shorter than 10 days, also in slide 33, it is possible that the employees allocated to task T1 do(es) not really use 10 person days (and 15 calendar days).

In the case of T1, it seems fine since in slide 32 we do see that T1 is from the beginning of week 0 till the beginning of week 2, so duration of 10 days is checked. In slide 33 we see that Jane is working full time from start of week 0 till start of week 2 for 10 days and Ali is working full time from start of week 0 till start of week 1 for 5 days, hence efforts 15 person days (15 = 10 + 5) is also checked.

**Make a table** to facilitate your answering (and my grading).

Table 1. Task scheduling and staff allocation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Effort | Duration | Dependency | Staff 1 | Staff 2 | Staff 3 | Remark |
| T1 | 15 | 10 |  | Jane 100% | Ali 100% |  | OK 31,32,33 |
| T2 | 8 | 15 |  |  |  |  |  |
| T3 | 20 | 15 | T1 (M1) |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Expand the table** to 12 rows for the 12 tasks in slide 31. **Add** columns or ways to clearly show me that you have checked all 12 tasks (with all the staff of 7 employees)

1. (3% - 10%) **Discuss** on if part (a) can be done by computer software (for project manager and for someone who comes to audit the project should there be 200 tasks, 30 months, 50+ employees, for a multimillions project)
2. (12+%) **Assume** T1 is delayed 3 days and T2 is delayed 2 days. Find out how many other tasks are delayed. How late will the project be delayed?
3. (8%) *Compute* the man-weeks effort of the original project (if an employee works 3 weeks, it is 3 man-weeks, 2 employees work 3 weeks, it is 6 man-weeks). *Compute* also the man-weeks of the delayed project assuming T1 is delayed 3 days and T2 is delayed 2 days (then many tasks but maybe not all tasks and milestones are affected)
4. (0% - 6%) **Discuss** the possibility of of doing part (c) and (d) by software.

**Bad Smells in Program Code (15%)**

1. (15%) Bad smells. In chapter 9 on evolution, there are two slides (#57 and #58) talking about bad programming practice like follows: in slide #57, duplicate code, long methods, switch statement; in slide #58, data clumping and speculative generality. A total of 5 different kinds of problems were explained in slides 57 and 58.

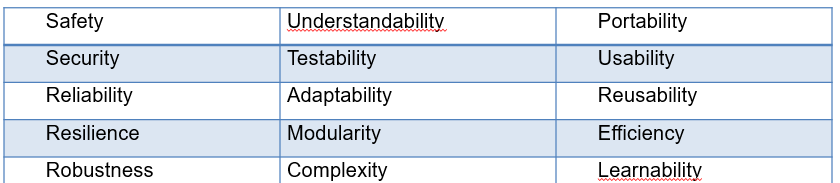
You had written code in report 2 (several programs), report 4 (email programming), report 5 (Excel VBA) and report 6 (Web programming).

1. (7%) No such problems in your code (or if there is, how would you fix). Explain briefly that you do NOT think your code in any of report 2, 4, 5, or 6 has such “bad smells”. It is impossible to include all your code in the answer here, but try to have one or two code snippets (of 5 to 10 lines) showing how you code and that you do NOT have such problems in the code snippet you explain here
2. (8 + %) **Examples of code** with “bad smells”. You can either search internet to find some programs that commit at least one of the bad smells problem or you can “make” one program that has at least one out of the 5 problems. Note the problematic source code should compile and run without logic errors. The only problem as mentioned in #57 and #58 should be called **style problem**. You can earn more than 8 points for this part if you find or create program with more than one problem
3. (5%+ \*). Discuss if there may be other “bad smells” that Sommerville did NOT mention in slides #9.57 and #9.58.

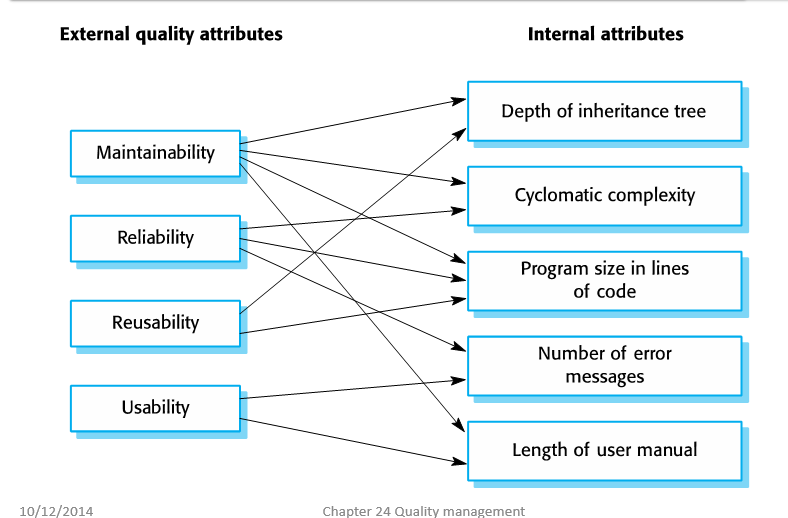
**Quality Attributes (15%)**

1. Quality Attributes. Chapter 24 of Sommerville discusses Quality Attributes (a software can be rejected by the customer due to poor quality).

In slide #24.13, 15 quality attributes were listed



In slide #54, external and internal quality attributes were mentioned



1. (7%) Discuss similar to part (a) of Q2 here the possible quality attributes of your programs (in report 2, 4, 5, and 6) or in your programs from the other classes, from the projects during your work etc. whether your code possesses any of the “good” quality attributes (the more the better) or in case none of the quality attributes in this chapter 24 applies to your code, make / create a software quality attribute and proceed with that.
2. **(8%+) Examples of programs with bad quality**. Similar to part (b) of Q2, find or make examples of software programs with bad quality where one or more of the quality attributes is bad.
3. (5%\* + ) Discuss if there may be any other quality attributes not mentioned in these slides of Sommerville including #24.13 and #24.15.

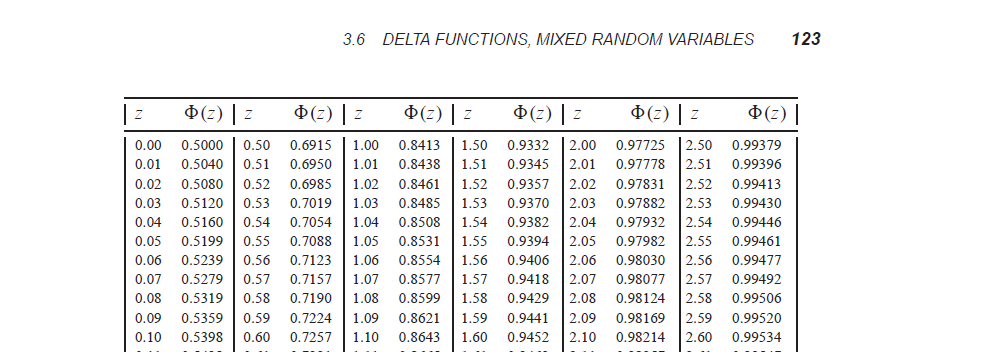
**Programming Questions (80%)**

**Generate Gaussian distribution table: 15% (ECE 3715)**

1. (15%) Gaussian Distribution table

Gaussian distribution is used a lot (normal distribution with the bell curve). Here is a table used in ECE 3715,

with the top of table captured as follows:



Please see the separate PDF file (in Blackboard) called Probability and Stochaistic Processes.pdf, chapter 3.5 from page 118 till page 122 of explanation of Gaussian distribution and table 3.1 on page 124 that presents the data of the normalized Gaussian distribution (with mean  = 0 and standard deviation  = 1, also labelled as Gaussian (0, 1)) for z = 0 until z = 2.99 in increments of 0.01.

1. (12%) You have learned Trapezoidal rule (or Simpson’s rule) in integral calculus (Mat 115). Using that rule plus C programming (or Python etc.) to **generate some table similar to table 3.1 from z = 0 till z = 4.99** (> 2.99) (table 3.1 in the book is till z = 2.99).
2. (3%) Verify your table agrees with table 3.1 from z = 0 till z = 2.99.

**Compute the roots of function: (15%)**

1. (15%) Consider the cubic polynomial function f(x) = x3 + x + 1. We know that this function is strictly increasing (i.e. if x1 < x2, then f(x1) < f(x2)).
2. (3%) Prove that this function is strictly increasing using differential calculus (Mat 114)
3. (12%) We know f(-1) = -1 – 1 + 1 = -1 < 0 and f(0) = 1 > 0. So from calculus, we know that there is a root (x0) such that f(x0) = 0 with -1 < x0 < 0.

**Write** a computer program in C / C++ / C# / Python to calculate this root to at least 5 digits precision (do NOT use MATLAB’s root command). You may use algorithm like bisection, Newton etc.

(you may use the PDF from Burden and Faires chapter 2. Pdf from EGR 5110 I instructed).

Class Polynomial (50%)

1. **(50%) Class Polynomial**

Polynomial is actually an array, but every entry is given a different meaning like the power of that entry. For example {1, 2, 3, 4} can mean x3 + 2x2 + 3x + 4 if we order them from the highest power down or 1 + 2x + 3x2 + 4 x3 if we start from the constant term.

Every polynomial also has a degree, the power of the highest power term. So a polynomial f(x) = x3 + 2x2 + 3x + 4 has a power of 3. Constant polynomials are usually assigned a degree of 0. (it is somehow controversial on whether number 0 is assigned degree 0 or minus infinity -∞. We will NOT discuss that here).

MATLAB is especially superior in handling polynomials since polynomials are just arrays, which are matrices (MATLAB stands for Matrices LAB).

1. (4%) Build a class myPolynomial starting with probably a double polynomial of degree 3.
2. (6%) Add the method degree for the polynomial object. Add also a method print or display so that you can print your polynomial on the screen.
3. (4%) Add the method Add to add a second polynomial. Note the sum of two polynomials can have lower degree since the highest degree term may cancel. Note deg (f + g) <= max (deg (f), deg (g))
4. (2%) Add the method Subtract (or Difference)
5. (8%) Add the method Multiply (or product) to compute the product of two polynomials. Note degree (fg) = deg (f) + deg (g). So the starting degree 3 and polynomial array of 3+1 = 4 entries may NOT suffice.
6. (10%) Add the method Division to divide polynomial A by polynomial B. Note we have quotient and remainder as you learned in high school (or kindergarten)
7. (16%) Test your results.

First, test that Multiply and Division work. So if you divide a(x) by b(x) and you get a(x) = b(x) \* q(x) + r(x) using your division method of class myPolynomial, then b(x) \* q(x) using your Multiply from part (e) and then b(x) \* q(x) + r(x) using your Add method from part (c), your b(x) \* q(x) + r(x) should be equal to a(x).

**Use for example a(x) corresponding to 4318 or array / polynomial {4, 3, 1, 8}** and b(x) corresponding to 480 or 3310.

**To test your Multiply, test with (1+x) 3 and (1+x) 10** (that we can verify using Pascal’s triangle).

Note (1+x) 3 just means you multiply 1+x three times using your method Multiply in part (e), so is (1+x) 10 (you probably need a for loop).