SENG 426: Midterm-Review Questions

Note: The midterm exam is open book and will be based on chapters 1-6 of the lecture notes. It is recommended to bring a calculator. You can print and bring your class notes but no computer is allowed.

Exercise 1: You are planning for the development of software for a system, which is very similar in size, complexity and requirements to a previous product used for online university course management. What estimated effects will the following process changes have on the percentage defect free for this new system?

1. You decide to do no inspections at all to save time and money. Why would this likely not save you time or money at all?
2. You suggest changes to your code inspection process based on problems you have identified in past projects. You expect these changes to have the effect of improving code inspection yield to 71.4%.

Exercise 2: To save costs the management of a software development company has instructed the QA department to conduct a thorough inspection process for their Inventory management software product, and then release the product, without undergoing a testing cycle. The software product consists of 100,000 lines of source code (100 KLOC). The defects data involved in this process are summarized in the following tables.

Development Phases	Requirements	High level design	Low level Design	Code
Defects Injected	122	859	939	1537

Verification Activities	Requirements Inspection	High level Design Inspection	Low level Design Inspection	Code Inspection
Defects Found	-	730	729	1095

The first table shows the development phases in which the defects are injected, while the second one shows the phases in which the defects are found (and removed). Compute the following metrics and comment on the results:

1. The phase effectiveness for the different inspection phases.
2. The overall Defect Removal Effectiveness (DRE)
3. The total number of defects remaining in the product at release time.
4. The defect density at release time.

Exercise 3: Consider a method that returns the number of days in a month, given the month and year.

```
class GregorianCalendar {
        //…
        public static int getNumDaysInMonth (int month, int year) {…}
        //…
}
```

The month and year are specified as integers. By convention, 1 represents the month of January, 2 the month of February, and so on. The range of valid inputs for the year is *0* to *maxInt*. Identify equivalence classes and boundary cases, and derive sample test cases accordingly.

Exercise 4: Derive a set of test cases for the following components:

1. A sort routine, which sorts arrays of integers.
2. A routine, which takes a line of text as input and counts the number of non-blank characters in that line.
3. A routine, which examines a line of text and replaces sequences of blank characters with a single character.
4. An object that implements variable length character strings. Operations should include concatenation, length (to give the length of a string) and substring selection.
5. An object representing a keyed table where entries are made and retrieved using some alphabetic key.

Exercise 5 (cf. Textbook Ex. 2 page 495): Assume that a software system will experience 150 failures in infinite time. The system has now experienced 60 failures so far. The initial failure intensity at the beginning of system testing was 15 failures per CPU hour. What is the current failure intensity?

Exercise 6 (cf. Textbook Ex. 3 page 495): Assume that a software system is undergoing system-level test and the initial failure intensity is 15 failures per CPU hour. The failure intensity decay parameter has been found to be 0.025 per failure. So far test engineers have observed 60 failures. What is the current failure intensity?

Exercise 7: Assume that a software system is undergoing system level testing. The initial failure intensity of the system was 25 failures/CPU hours, and the current failure intensity is 5 failures/CPU hour. It has been decided by the project manager that the system will be released only after the system reaches a reliability level of at most 0.001 failures/CPU hour. From their experience the management team estimates that the system will experience a total of 1200 failures over infinite time. Calculate the additional length of system testing required before the system can be released.

Exercise 8:
You are now in system test for an online Theater Tickets purchasing software. You have the defect arrival data points below. Assume a Raleigh curve.

a. What do you predict as the total number of bugs in the system? Use the 40% method (i.e. method 1).
b. How many bugs do you predict as being left in the system?
c. What is the equation that predicts the defects?
d. If you shipped at the end of week 6 (and assuming you removed all the defects found at that time), what would you predict as the defect removal efficiency?
e. If this is a 5,000 LOC program, what would you predict as the remaining defect density after 6 Months?
f. Should you ship after 6 Months? Why or why not?

Month found	1	2	3	4	5	6
Number of defects	13	22	25	22	17	5

Exercise 9 (4.4 in the slides- Unit #4):

It has been estimated that the safety-critical subsystem of a patient monitoring system will experience a total of $\nu_0 = 120$ failures (in infinite time). Suppose that healthcare regulations require a failure intensity of $\lambda_{obj}= 0.001$ failures/CPU hr for such critical component before the product could be released.

1. Considering that the testing starts with an initial failure intensity of $\lambda_0 = 20$ failures/CPU hr, calculate the number of failures and the amount of execution time required to reach the failure intensity objective.

The effort required per hr of execution time is 6 person hr (for failure identification). Each failure requires (additionally) 2 person hr on the average to verify and determine its nature.

2. Calculate the total cost of testing assuming a loaded salary of $40/hr and additional cost of $50/hr for resources (e.g. Computer) and overheads.
3. Calculate the total duration of the testing activity in calendar time, assuming a standard work week of 40 hrs, and a 2 members (full time) test team.