SENG 426: Midterm-Review Solutions

Exercise 1:

1.

Studies conducted by IBM over several major projects have shown that:

- Number of major defects could be detected during application development
- 50% of the defects could be detected when testing is conducted before the implementation phase,
- 80% could be detected when testing is conducted after implementation.
- It is at least 10 times more expensive to fix a defect after implementation than before, and 100 times more expensive during maintenance.

Hence, because the cost of fixing defects increase significantly as we move along the development cycle, this option might turn to be far less cost-effective. Defects discovered are more expensive and time consuming to be fixed.

2. There is a great chance that you reach your goal of improving the quality of your product, because process change is based on historical data from past similar project.

Exercise 2:

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

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

1.

There were no formal requirements inspection, so we are unable to assess the effectiveness of this phase; defects injected in this phase would be found later in the development cycle.

High-level design inspection:
Defects removed: 730
Defects existing on step entry (escapes from requirements phase): 122
Defects injected in current phase: 859

PE (HLD) = 730/(122+859) = 74%

Low-level design inspection:
Defects removed: 729

Defects existing on step entry (escapes from requirements and HLD phases): 122 + 859 – 730 = 251
Defects injected in current phase: 939

PE (LLD) = 729/(939 + 251) = 61%

Code inspection:
Defects removed: 1095
Defects existing on step entry: 122 + 859 +939 – 730 – 729 = 461
Defects injected in current phase: 1537

PE (CI) = 1095/(461 + 1537) = 55%

2. Overall DRE:

DRE = (730 + 729 + 1095)/(122 + 859 + 939 + 1537) = 74%

3. Number of defects remaining:
n= (122 + 859 + 939 + 1537) - (730 + 729 + 1095) = 3457-2554 = 903

4. Defect density:
DD=9.03 defects/KLOC

The system will perform poorly if released at this stage. Recommend strongly the management to conduct further testing.

Exercise 3:
For the month parameter, there are 5 equivalence classes:

1. Months with 31 days (i.e., 1, 3, 5, 7, 8, 10, 12),
2. Months with 30 days (i.e., 4, 6, 9, 11)
3. February (i.e., 2), which can have 28 or 29 days.
4. Non-positive or null integers (month ≤ 0), which are invalid
5. Integers larger than 12 (12 < month), which are also invalid.

For the year parameters, there are 4 equivalence classes:

6. Non-leap years
7. Leap years
8. Negative integers (year < 0) are invalid values.
9. Integers above maxInt (maxInt < year) are also invalid values.

In general, years that are multiple of 4 are leap years, in which case the month February includes 29 days instead of 28 (for non-leap years). Years that are multiple of 100, however are not leap years unless they are also multiple of 400. For example, 2000 is a leap year whereas 1900 was not.

To generate valid test inputs, we select one valid value for each parameter and equivalence class (e.g., February, June, July, 1901, and 1904). Combining the parameters actually yields 6 equivalence test cases (test cases 1-6 for classes 1, 2, 3, 6, and 7).

Test case #	Equivalence class	Value for month input	Value for year input	Expected output
1	1, 6	7	1901	31
2	1, 7	7	1904	31
3	2, 6	6	1901	30
4	2, 7	6	1904	30
5	3, 6	2	1901	28
6	3, 7	2	1904	29
7	4	-1	2000	Invalid
8	5	13	1950	Invalid
9	8	3	-1	Invalid
10	9	10	MaxInt+1	Invalid

There are four boundary cases; the following table lists boundary test cases:

Test case #	Equivalence class	Value for month input	Value for year input	Expected output
11		1	1291	31
12		12	1315	31
13		10	1	31
14		11	maxInt	30

Exercise 4:

1. Sort routine

No	Category	Input	Expected output
1	Empty array	[]	[]
2	Singleton	[2]	[2]
3	Ordered array	[3,4,10,15,16]	[3,4,10,15,16]
4	Disordered array	[1, 10,4, 2, 5, 15]	[1,2,4, 5,10,15]

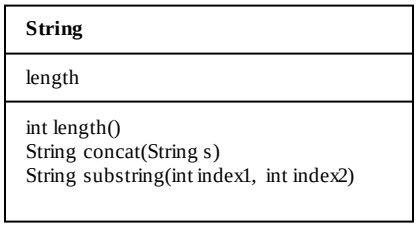
2.Non-blank character counter

No.	Category	Input	Expected output
1	Empty line	“ ”	0
2	Single word	“john”	4
3	Multi word line	“john goes to school”	16

3. Blank editor

No.	Category	Input	Expected output
1	Empty line	“ ”	“ “
2	Regular line	“john goes to school”	“john goes to school”
3	Line with space	“john goes to school”	“john goes to school”

4. String object



Length operation:

- Empty string-> output =0
- One character string -> output = 1
- More than one character: “john” -> 4

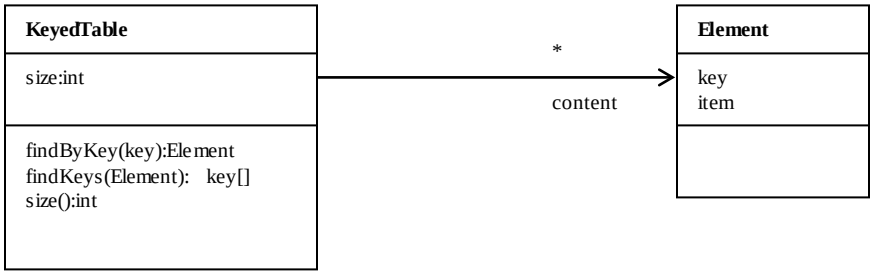
Concat operation:

- Two empty string
- An empty string and a non-empty string
- Two regular strings

Substring operation:
The following condition applies:
Length $\neq 0$ => $1 \leq index1 \leq index2 \leq length$
Length $= 0$ => $index1 = index2 = 0$

No.	Category	String object	Index 1	Index 2	Expected output
1	Empty string	“”	0	0	“”
2	Empty string	“”	0	2	invalid
3	Empty string	“”	1	0	invalid
4	Single character	“a”	1	1	“a”
5	Index1 <1 and length $\neq 0$	“john”	0	3	invalid
6	Index1 = 1	“paul”	0	2	“pa”
7	Index1 = index2	“jonas”	3	3	“n”
8	Index1 > index2	“paolo”	3	2	Invalid
9	Index2 = length	“paolo”	2	5	“aolo”
10	Index2 > length	“paolo”	2	6	invalid

5. Keyed table



Test may be organized around the following cases:

- Empty table
- Singleton
- Multi-element table

This may lead, for instance, to the following cases:

- findByKey: a case with a valid and a case with an invalid key for multi-element
- size(): empty, singleton, and multi-element
- findKeys: a case with a valid element, a case with an invalid element, a case with a valid element stored in more than one location.

Exercise 5:
We assume in this exercise that the failure occurrence is based on the basic execution model.
The total number of failures (in infinite time) is ν_0=150 failures
The initial failure intensity at the beginning of testing is λ_0= 15 failures/CPU hr

The system has experienced at this stage (of testing) $\Delta\mu$ = 60 failures.
The current failure intensity λ_1 can be calculated as follows (using the basic execution model):

$$\Delta\mu = \frac{\nu_0}{\lambda_0}(\lambda_0 - \lambda_1) =$$
$$\Rightarrow$$
$$\lambda_1 = -\frac{\lambda_0}{\nu_0}\Delta\mu + \lambda_0 = -\frac{15}{150} \times 60 + 15 = 9 \text{ failures/CPU hr}$$

Exercise 6:
We assume in this exercise that the failure occurrence is based on the basic logarithmic model (because the decay parameter is provided).
The failure intensity decay parameter is Θ=0.025/failure
The initial failure intensity at the beginning of testing is λ_0= 15 failures/CPU hr

The system has experienced at this stage (of testing) $\Delta\mu$ = 60 failures.
The current failure intensity λ_1 can be calculated as follows (using the logarithmic model):

$$\lambda_1 = \lambda_0 e^{-\mu\theta} = 15 \times e^{-60\times 0.025} = 3.35 \text{ failures/CPU hr}$$

Exercise 7:
We assume in this exercise that the failure occurrence is based on the basic execution model.
The total number of failures (in infinite time) is ν_0=1200 failures
The initial failure intensity at the beginning of testing is λ_0= 25 failures/CPU hr.

The current failure intensity is λ_1= 5 failures/CPU hr

The failure intensity objective before releasing the system is λ_{obj} =0.001 failures/CPU hour.

The additional time of testing to reach the failure intensity from the current stage can be calculated (using the basic execution model) as follows:

$$\Delta\tau = \frac{\nu_0}{\lambda_0} ln\left(\frac{\lambda_1}{\lambda_{obj}}\right) = \frac{1200}{25} \times ln\left(\frac{5}{0.001}\right) = 408.82 \text{ CPU hr}$$

Exercise 8:

a.

The maximum defect arrival occurs at tm=3rd month.

Using the 40% rule: tm= 3 => F(tm) = 13 + 22 + 25 = 60 => K = 60/0.4 = 150 defects

b.

With release at month 6: defects remaining D = 150 -104 = 46 defects

c.

With K=150 and c= tm x sqrt (2) = 4.24, we get the following equations:

$$F(t) = 150\left(1 - e^{-\left(\frac{t}{4.24}\right)^2}\right) = 150 \times \left(1 - e^{-0.055 \times t^2}\right)$$

$$f(t) = K\left(\frac{2t}{c^2}\right)e^{-\left(\frac{t}{c}\right)^2} = 150 \times \left(\frac{2t}{4.24^2}\right)e^{-\frac{t^2}{4.24^2}} = 16.68 \times e^{-0.055 \times t^2}$$

d.

DRE = 104/150 =69.33%

e.

Residual defect density = 46/5KLOC= 9.2 defects/KLOC

f.

The residual defect density is still very high; so a no ship decision will be appropriate. The best course of action in this case will be to extend the testing period.

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

1. $\Delta\mu$=(120/20)x(20-0.001)=120 failures

$\Delta\tau$=(120/20)xln(20/0.001)=59.42 CPU hr

2. Total failure identification effort:
   Effort = id + analysis = 6x59.42 + 2x120 = 596.52 person hr

=> Cost = personnel + resources = 40x596.52 + 50x59.42 = $26,831.8

3. Time = Effort/2= 596.52/2=298.26 hrs = 7 weeks 3 days