Sample Exams

These sample exams are handed out for two reasons: 1) to give you a taste of the style of questions that may appear on the exam; and 2) to give you a practice run. They are by no means meant to be taken as a hint to the questions that will be on the exam, nor should they be used as a study guide. This document should be used in conjunction with the subject notes, the in-lecture exercises, and the tutorials.

It is recommended that your first attempt for at least one of these sample exams should be done as if it were an actual exam. That is, sit at a table with the same materials that you will take into the exam, and try to complete this within the two hours.

These exams, as well as the final exams, contain 100 marks. The final exam is worth 50% of your final mark, so each mark on the exam is worth 0.5%. The number of marks assigned to each question should be an indicator of the amount of time you should spend on each question. My recommended strategy is to spend 1 minute per mark, which for a 2 hour exam, gives you 20 minutes at the end to check your answers.

There will be no sample solutions handed out. The reasons for this are quite simple. First, there are no solutions handed out with the actual exam. Having solutions handy encourages people to peek at them while they are doing the questions, which is not good practice for the exam at all. Second, many people tend to use the exam answers as a study guide. This is not a sound way to study, so we want to discourage this.

We encourage students to post their sample solutions to the discussion board for others to see and comment on. If the subject staff think that there is value on commenting on particular answers, they will contribute too.

The University of Melbourne

SWEN90006: Software Testing and Reliability

Sample Exam 1

Semester 2, 2016

Part A – Functional Testing

Question 1 [8 marks]

(i) **[4 marks]** Compare and contrast active test oracles with passive test oracles. Your answer should make reference to strengths of each approach and the weaknesses of each approach.

(ii) **[4 marks]** The mantra "design for testability" refers to the notion that, when designing software, we should consider the factors of controllability and observability. How do these these two terms related to testing, and what is there importance?

Question 2 [20 marks]

This question refers to the following specification.

> Consider a simple database for storing client account details that users can access in the field by using a mobile phone. To access the database users must first enter one of two phone numbers to dial into the database. Once they are connected they must enter a # followed by a password followed by another #. Then users can access customer accounts by typing in the account code and the data that they wish to see. Once all of the data is entered the system responds with the requested information if the password and fields are valid.
>
> The following is the valid data for the protocol just described where the *send* command is assumed to be part of the mobile phone system and not part of the system under test..
>
> **Phone Numbers** – The two legal phone numbers are 555-5975 and 555-4976.
>
> **Password** – The password must be a hash (#) followed by a six (6) digit number followed by a hash (#) followed by a *send*.
>
> **Account Codes** – An account code is a three digit number in the range 100-499 inclusive followed by a *send*.
>
> **Fields** – The fields are selected by entering a three digit string consisting of just 1's and 0's which are interpreted as follows:
>
> - 001 – Client name;
> - 011 – Client address;
> - 100 – Current balance.
>
> The user must perform a *send* once this data is entered as well.

1. [**4 marks**] Explain the equivalence partitioning strategy for selecting test inputs.

2. [**6 marks**] Specify the input conditions for the mobile phone database application described at the start of this question.

3. [**10 marks**] Use equivalence partitioning to derive
   (i) equivalence classes for the input conditions; and
   (ii) test inputs for the mobile phone database application described above.

Question 3 [10 marks]

1. [**3 marks**] With respect to data-flow analysis, in a sentence or two, state what each of the following data-flow anomalies may indicate:
   (i) a `d-d` anomaly;
   (ii) a `u-r` anomaly; and
   (iii) a `d-u` anomaly.

   Note: this question is not asking you to define the above terms, but to explain why anomalies of this type may indicate a problem a program.

2. [**7 marks**] The program fragment in Figure 1 replaces a substring of `a` by the string `p` starting at the index `j`.
   (i) Give two data flow anomalies for the program in Figure 1.
   (ii) What does each of your two data-flow anomalies indicate about the program?

```
void string_insertion(char * a, char * p, int j)
{
    int i, length_p, length_a;

    /* determine the length of p and a */
    length_p = strlen(p);
    length = strlen(a);

    /* replace the substring of a starting at j by p */
    if (0 <= j && j < length_a && j + length_p < length_a) {
        for (j = 0; i < length_p; i++) {
            a[j+i] = p[i];
        }
    }

    /* print it! */
    printf("%s\n", a);
}
```

Figure 1: A string insertion program.

Question 4 [20 marks]

The program in Figure 2 takes a single string in the array `text` and partitions it into an array of lines stored in `page`. Each time a newline character `'\n'` is reached a new line is created.

```
void split(char * text, char ** page)
{
    int i, line, place;

    line  = 0;
    place = 0;

    for (i = 0; i < strlen(text) - 1 && text[i] != 0; i++) {
        if (text[i] == '\n') {
            line = line + 1;
            place = 0;
        }

        page[line][place] = text[i];

        if (text[i] != '\n') {
            place = place + 1;
        }
    }
}
```

Figure 2: The `split` program to split a single string into a set of lines.

1. [**4 marks**] Briefly explain the aim of the condition coverage test input selection strategy.
2. [**6 marks**] Draw a control-flow graph for the program in Figure 2.
3. [**2 marks**] List the conditions in the program of Figure 2.
4. [**8 marks**] Specify a set of test inputs that achieves branch coverage of your control-flow graph. Briefly explain how the inputs cover the branches.

Question 5 [10 marks]

1. [**4 marks**] Consider a single mutant that has been generated for a program that we are testing. We have a test suite that reveals no failures in the original program. This test suite is run on the mutant, and again, no failures occur. What does this mean about our test suite?
2. [**6 marks**] The *relational operator replacement* rule takes an occurrence of a relational operator, $<, \leq, >, \geq, =$, or $\neq$, replaces that occurrence with one of every other type of relational operator to generate a set of mutants. A colleague in your quality assurance team hypothesises that boundary-value analysis is required if we want to kill all non-equivalent mutants created by this operator. Do you agree with your colleague? Explain why.

Part B – Software Reliability Modelling

Question 6 [6 marks]

Explain what is meant by "software reliability". Your answer should define software reliability, and explain how the interpretation of failures affect the estimates of failure intensity.

Question 7 [12 marks]

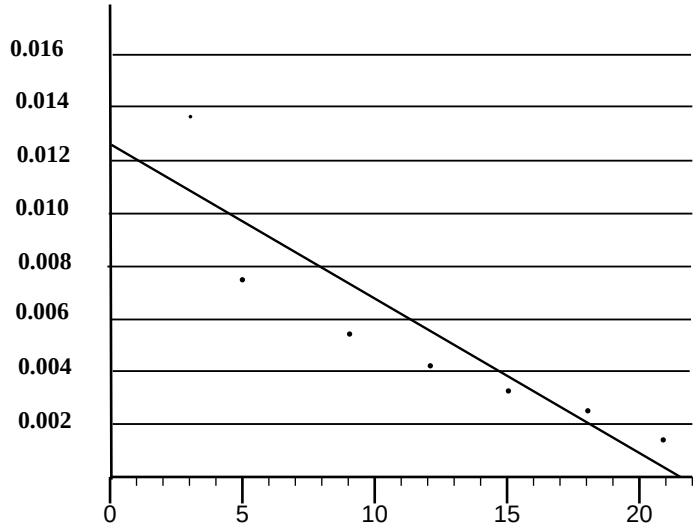

Figure 3: Graph of failure intensity against mean no. of failures experienced.

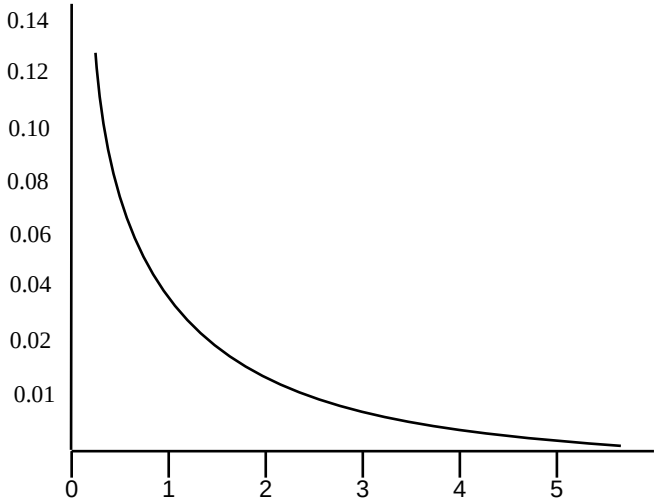

Figure 4: Graph of failure intensity against execution time.

Suppose that you are in charge of the *Absolutely Safe Aeroplane Flight Control System* project and that your testing manager arrives at your office door one morning and shows you the graphs in Figure 3 and Figure 4. The regression equation in Figure 3 is

$$\lambda(\mu) = -0.00574\mu + 0.124153$$

with the estimate of the total number of failures $\nu_0 = 21.6952$.

1. **[3 marks]** What effect does the first data point in Figure 3 at $\mu = 3$ and $\lambda = 0.136364$ have on the initial failure intensity λ_0 and the estimate of the total number of failures ν_0?

2. **[9 marks]** The testing manager asserts that to date, the testing team have experienced and repaired fourteen (14) failures during random testing. Consequently, if the testing team experiences a further seven (7) failures then the system will have a failure intensity of 0, that is, it will be failure free!

   Do you agree with this statement and what would your response to the testing manager be? Your answer must make reference to:

   (i) the accuracy of the the estimates for the initial failure intensity λ_0 and the total number of failures ν_0;
   (ii) the meaning of the *slope* of the regression line; and
   (iii) the dependence of the testing results on the *operational profile*.

Part C – Symbolic verification

Question 8 [14 marks]

Consider the following program:

```
int foo(int x)
{
  int result = 1;
  if (x < 0) result - x;
  if (x > 0) result + x;
  return result;
}
```

1. **[10 marks]** Using the theory of symbolic execution, calculate all possible symbolic states/outputs for the foo program. To answer your question, construct a symbolic execution tree for the program. As part of your symbolic execution tree, use the variable $\mathtt{X}_0$ to represent the initial value of the variable `x`.

2. **[4 marks]** An *integer overflow* occurs in a program when an arithmetic creates a value that is too large to fit within the available storage space. For example, on a 32-bit architecture, any number smaller than $2^{31} + 1$ or larger than $2^{31} - 1$ cannot be represented.

   Use your symbolic execution tree to demonstrate whether there is an integer overflow in the `foo` program.

The University of Melbourne

SWEN90006: Software Testing and Reliability

Sample Exam 2

Semester 2, 2016

Part A – Functional Testing

The word count program counts the number of lines, words and characters in an input string. The requirements for `WordCount` are given below.

> The `WordCount` program accepts a string from the standard input and delivers a count of the number of lines, words and characters in the input string.
>
> (i) The program should accept a string of up to 100 words in length.
> (ii) Words are separated by white spaces. White spaces are the space character, tabs, and new line characters.
> (iii) The output is three integers as follows:
>
> L W C
>
> Where L, W, and C are integers that are equal to the number of lines, words and characters respectively in the input string.

Question 1 [8 marks]

Answer the following questions about the `WordCount` program:

(i) **[4 marks]** Determine the input and output domains for the `WordCount` program; and

(ii) **[4 marks]** Determine the input and output conditions for the `WordCount` program.

Question 2 [12 marks]

Based on your analysis in question 1 above, generate a set of test cases for the `WordCount` program using equivalence partitioning. Justify your choices of test cases from the equivalence partitioning guidelines where relevant.

An implementation of the `WordCount` program, written in C, is given in Figure 1. Note, there are two seeded errors.

Question 3 [14 marks]

Draw the control-flow graph for the `WordCount` program. You may break the function up into basic blocks to simplify your CFG.

```
#define NOT_IN 0;           // not inside a word

int main(void) {
    int wordState = NOT_IN; // current word state
    int numChars = 0;       // characters so far
    int numWords = 0;       // words so far
    int numLines = 0;       // lines so far

    ch = getchar();
    while (ch != EOF) {
        numChars++;
        if (ch == '\n') {
            numLines++;
        }

        if (wordState == NOT_IN) {
            if (!isspace(ch)) {
                numWords++;
                wordState = NOT_IN;
            }
            else {
               if (isspace(ch)) {
                  wordState = NOT_IN;
               }
            }
        }
        ch = getchar();
    }
    printf("%d %d %d \n", numLines, numWords, numChars);
    return 0;
}
```

Figure 1: C implementation for the `WordCount` program.

Question 4 [10 marks]

Derive a set of test inputs for the `WordCount` program to meet the branch coverage criteria. Show which branches are exercised by your choice of test inputs.

Question 5 [2 marks]

Derive a set of test inputs for the `WordCount` program to meet the multiple-condition coverage criteria. How do these differ from your test inputs that achieve branch coverage?

```
public class Queue
{
  /* Queue Constructor */
  Queue(int size) {...}

  /* Add a value to the end of the queue. */
  public void add(Object value) {...}

  /* Remove a value from the head of the queue. */
  public Object remove() {...}

  /* get the value at the head of the queue. */
  public Object first() {...}

  /* Determines if the queue is not able to accept any new values. */
  public boolean isFull() {...}

  /* Determine if the queue is empty. */
  public boolean isEmpty() {...}
}
```

Figure 2: An interface for a `Queue` class in Java.

The follow questions refer to the Queue ADT given in Figure 2.

Question 6 [14 marks]

Draw a state transition diagram for the Java class in Figure 2. Clearly label all of the your transitions, guards on transitions and states. You can use a key or legend at the side of the diagram once you have worked out your states.

Question 7 [12 marks]

Using your state transition diagram, generate a sample of four test cases for the `Queue` class. With reference to your state transition diagram, what states in the diagram do each of your test cases exercise?

Part B – Software Reliability

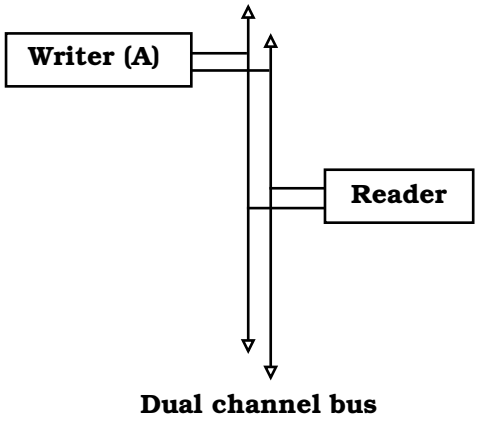

Figure 3: A single Reader/Writer system communicating over a dual channel bus.

Question 8 [14 marks]

The writer communicates with the reader via a two channel bus. Of concern to developers in the Reader/Writer system in Figure 3 is the reliability of the dual channel bus.

The failure and repair probabilities, as measured, for each of the two channels are given in the following table:

	Failure Probability	**Repair Probability**
Channel A	0.001	0.9
Channel B	0.002	0.8

(i) **[6 marks]** Draw the Markov state transition diagram showing all of the probabilities.

(ii) **[2 marks]** Give the transition matrix for the two channel bus.

(iii) **[6 marks]** Sketch a curve of how you would expect the failure probability to change with *time* for the for the two channel bus. NOTE: you do not need to calculate the n-Step transition matrix, just provide the general shape of the curve.

Question 9 [14 marks]

Consider the graph in Figure 4, which maps the relationship between failure intensity and number of faults found for a project. The graph is obtained from failure intensity estimates using the basic execution time model. The regression line for the graph in Figure 4 is

$$\lambda(\mu) = -0.00574\mu + 0.124153.$$

Answer the following questions:

(i) **[6 marks]** What does it mean for a system to be "reliable", and how can we assess reliability?

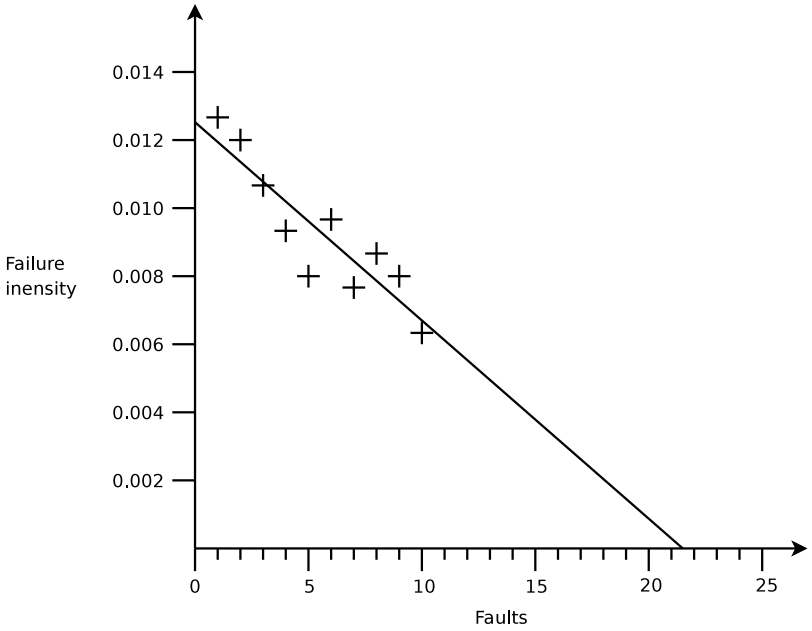

Figure 4: Reliability graph for a project after finding 10 faults.

(ii) **[4 marks]** Using *only* the graph in Figure 4, provide a rough estimate of the total number of expected faults that were in the system at the start of reliability testing. How did you derive this value?

(iii) **[4 marks]** Using your estimate of the total number of faults, show the expression that you would use to calculate the total number of failures at a time t. Note: there should be no need to use your calculator for this: just give the expression that you could use to calculate this.

Sample Security Questions

These questions are included separately since much of the security testing material is new this year, so wasn't represented on prior exams.

The University of Melbourne

SWEN90006: Software Testing and Reliability

Sample Security Questions

Semester 2, 2017

Part A – Security Testing

Question 1 [25 marks]

(i) **[4 marks]** Suppose you were conducting security testing on a server that processed raw network packets received from its clients. Which security testing technique would you use and why?

(ii) **[4 marks]** You are fuzzing a program to try to find security faults. What are the advantages and disadvantages of running the program under a memory debugger when fuzzing it?

(iii) **[10 marks]** The following function takes a string as its first argument an a length (integer) n as its second argument. Its job is to print the first n characters of the string. Imagine that the integer n is supplied by a potential attacker and that the program also holds some secrets elsewhere in memory.

```
void print_string(char* string, unsigned int len){
  for (unsigned int i=0; i<len; i++){
    putchar(string[i]); /* print the character */
  }
}
```

This program has a security fault. What is it and how can an attacker trigger it? (3 marks)

What are the potential security implications of this fault? (i.e. if an attacker were to exploit it, what might the attacker be able to cause to happen?) (3 marks)

What security testing strategy would you use to find this fault? (4 marks)

(iv) **[2 marks]** Consider an encryption function whose purpose is to take a string and a secret encryption key and to return the string encrypted under the secret key. If an attacker doesn't know the encryption key then they shouldn't be able to work out the original string if they are given the encrypted string. Imagine that an attacker can observe how long the encryption function takes to execute and suppose that its running time can be influenced by the string being encrypted and the secret key. What kind of covert channel does the encryption function have?

(v) **[5 marks]** General purpose greybox fuzzers like AFL or libFuzzer monitor the execution of the program under test and use this information to attempt to generate new program inputs that cause the program under test to execute paths that haven't already been explored during fuzzing.

Compare and contrast this general purpose greybox fuzzing with dynamic symbolic execution.