DIT635 - Final Exam
Name:

There are a total of 9 questions and 100 points available on the test. On all essay type questions, you will receive points based on the quality of the answer - not the quantity.

You may answer either in this document or in a separate document.

Question 1 (Warm Up) - 10 Points

Multiple solutions may apply. Select all that are applicable.

1. For the expression `(a || c) && (b || !c)`, the test suite **(a, b, c) = {(T, F, T), (F, F, T), (F, T, T), (T, F, F)}** provides:
    a. MC/DC Coverage
    b. Decision Coverage
    c. Basic Condition Coverage
    d. Compound Condition Coverage

2. Acceptance testing is a critical validation activity.
    a. True
    b. False

3. All DU pairs coverage requires that all paths between each definition and each of its usages be covered by at least one test case.
    a. True
    b. False

4. Any program that has passed all test cases and has been released to the public is considered which of the following:
    a. Correct with respect to its specification.
    b. Safe to operate.
    c. Robust in the presence of exceptional conditions.
    d. Considered to have passed verification.

5. In random ascent, we take the first neighbouring solution to show any improvement over the current solution as the new solution.
    a. True
    b. False

6. A mutant is considered valid - but not useful - if it compiles but the majority of tests fail.
    a. True
    b. False

7. You are designing an autopilot system for aircraft. You have designed the system to resend instructions if no acknowledgement is received, but it will cease after 10 attempts. What reliability measures would be of most interest to you?
    a. **Availability**
    b. Probability of failure on demand
    c. **Mean time between failures**
    d. **Rate of fault occurrence**

Question 2 (Quality Scenarios) - 10 Points

Consider an online shopping site, like Elgiganten. This site allows users to search and browse items, add items to a card, purchase items, and save items to a list tied to their account. Users have a persistent account with their personal information (i.e., name, address, e-mail address, order history, etc.). This site must enable secure purchases of items to thousands of concurrent visitors.

Create one performance and one availability scenario for this system, with a Description, System State, System Environment, External Stimulus, Required System Response, and Response Measure for each.

Question 3 (Testing Concepts) - 10 Points

Choose one stage of testing (unit testing, system testing, GUI testing, exploratory testing, or acceptance testing). Explain in your own words what that stage is, how systems are tested in that stage, how it differs from the other listed stages of testing, and the types of faults that are most likely to be exposed by that stage (that would be missed during the other stages).

Question 4 (System Testing) - 12 Points

Consider a method that recursively iterates over all files and subdirectories starting from a specified directory, searching for a stated file extension, and returns an array of all files with that extension:

```
String[] getFileList (String directory, String fileExtension)
```

Each file in the returned array is listed with its full path. All files with the stated extension in the stated directory of any of its subdirectories are contained in the list. If you wish to make any additional assumptions about the functionality of this method, state them in your answer.

Perform category-partition testing for this feature.

1. For each parameter, identify testing choices (controllable items that can be varied when testing)

2. Identify representative input values (types of input choices) for each choice.
3. Apply constraints (IF, ERROR, SINGLE) where they make sense.

You do not need to create test specifications or concrete test cases.

Hint: Do not forget about environmental factors that can influence system output.

For invalid input, **do not** just write “invalid” - be specific.

Question 5 (Exploratory Testing) - 8 Points

Exploratory testing typically is guided by “tours”. Each tour describes a different way of thinking about the system-under-test and prescribes how the tester should act when they explore the functionality of the system.

1. Describe one of the tours that we discussed in class OTHER than the supermodel tour.
2. Consider a web-based education management system like LADOK, where one can register for courses, see past courses taken, apply for credits or graduation, manage their student information, apply for a university, or transfer to a new university. Describe three actions you might take during exploratory testing of this system, based on the tour you described above.

Question 6 (Unit Testing) - 8 Points

Consider a Java method that accepts a string of comma-separated numbers and returns the sum of those numbers:

```
int convertToSum (String numbers) throws IllegalFormatException;
```

For example, `convertToSum(“1,3,45,1”)` returns the integer 50.

Write JUnit-format test cases to do the following:

1. Check that a well-formatted string returns the expected integer.
2. Check that a string containing illegal characters results in the method throwing an `IllegalFormatException` (for example, “1,HELLO,17” is illegal because “HELLO” is not a number.

Question 7 (Structural Testing) - 15 Points

For the following function:

```
1.  public static int jumpSearch(int[] nums, int item) {
2.      int array_size = nums.length;
3.      // Find block size to be jumped
4.      int block_size = (int)Math.floor(Math.sqrt(array_size));
5.      // If element is present find the block where element is present
6.      int prev_item = 0;
7.      while (nums[Math.min(block_size, array_size)-1] < item){
8.        prev_item = block_size;
9.        block_size += (int)Math.floor(Math.sqrt(array_size));
10.       if (prev_item >= array_size)
11.         return -1;
12.     }
13.     // Using a linear search for element in block beginning with
    previous item
14.     while (nums[prev_item] < item){
15.       prev_item++;
16.       if (prev_item == Math.min(block_size, array_size))
17.         return -1;
18.     }
19.     // If element is found
20.     if (nums[prev_item] == item)
21.       return prev_item;
22.     return -1;
23.   }
```

1. Draw the control-flow graph for this program. You may refer to line numbers instead of writing the full code.
2. Develop test input that will provide statement and branch coverage. For each test input, list the line numbers of the statements covered as well as the branches covered (use the line number and T or F, i.e., “7-T” for the true branch of line 7).
3. Do your test cases also achieve path coverage? Briefly explain your answer.

1. **(get from photos)**
2. **(1,2,3,17), 3**
   Covers 1-10,12-14,18-21,23
   7-T, 7-F, 10-F, 14-F, 20-T
   Missing 10-T, 14-T, 16-T, 16-F, 20-F
   (1,2,3,17), 20
   Adds 15-17, 14-T, 16-T
   Missing 10-T, 16-F, 20-F
   (1,2,3,17), 17
   Adds 15-16, 16F
   Missing 10-T, 20-F

(1), 2
Adds 11, 10-T
Missing 20-F
(2), 1
Adds 22, 20-F

3. **No - loops**

Question 8 (Data Flow Testing) - 12 Points

Consider the following Java method to check or find if a three-digit number is an Armstrong number or not (an Armstrong number of three digits is a number whose sum of cubes of its digit is equal to its number. For example, 153 is an Armstrong number because 1^3+5^3+3^3 or 1+125+27=153).

```
1. private static boolean isArmStrong(int number) {
2.     int result = 0;
3.     int orig = number;
4.     while(number != 0){
5.         int remainder = number%10;
6.         result = result + remainder*remainder*remainder;
7.         number = number/10;
8.     }
9.     //number is Armstrong return true
10.    if(orig == result){
11.        return true;
12.    }
13.    return false;
14.   }
```

1. Identify the def-use pairs for all variables.
   number: (1,3), (1,4), (1,5), (1,7), (7,4), (7,5), (7,7)
   result: (2,6), (6,6), (6,10), (2,10)
   orig: (3,10)
   remain: (5,6)
2. Provide a test suite that achieves all def-use pairs coverage.
   Will need a test that skips the loop to cover (2,10) pair for result. This requires a number that is 0.

Question 9 (Mutation Testing) - 15 Points

This function computes the longest common sequence of characters between two strings:

```
1. public String findLongestCommonSequence(String s1, String s2) {
2.     String result = "";
3.     for (int length = s1.length(); length > 0; length--) {
4.         int startIndex = 0;
5.         while (startIndex + length <= s1.length()) {
6.             String current = s1.substring(startIndex, startIndex + length);
7.             if (s2.contains(current)) {
8.                 result = current;
9.                 break;
10.             }
11.             startIndex++;
12.         }
13.         if (result.length() != 0) {
14.             break;
15.         }
16.     }
17.     return result;
18. }
```

Answer the following three questions for **each** of the following mutation operators:

- Relational operator replacement (ror)
- Arithmetic operator replacement (aor)
- Constant for constant replacement (crp)

1. Identify all lines that can be mutated using that operator. **(ROR - 3,5,13 ; AOR - 3, 5, 6, 11 ; CRP - 2, 3, 4, 13)**
2. Choose **one** line that can be mutated by that operator and create **one** non-equivalent mutant.
3. For that mutant, provide test input that would detect the mutant. Show how the output (return value of the method) differs from that of the original program.