Term Test 1

COSC 4313 3.0 Software Engineering Testing

Section M, Winter 2007

Family Name: _____

Given Name(s): _____

Student Number: |___ |___ |___ |___ |___ |___ |___ |___ |___ |

Question	Out of	Mark
Q1	20	
Q2	20	
Q3	20	
Q4	20	
Q5	20	
Total	100	
Letter grade		

1. **[20 marks]** In Open Office, pressing CTRL-DELETE is supposed to delete all characters until the end of the current word (words are separated by spaces or tabs). Following is a bug report related to this feature. Describe four different follow-up tests that you would like to run while analyzing this bug.

   Summary: CTRL-DELETE deletes two words if cursor is right after a dash
   Operating system: Windows XP
   Open Office version: 2.1.0
   Reproduction steps:

   (a) Start with a new text document in Open Office Writer
   (b) Type the following: `abc-def 123 sdf`
   (c) Position the cursor right after the dash.
   (d) Press CTRL-DELETE.

   The screen will now contain the following: `abc- sdf`

   Any of the following follow-up tests would be worth 5 marks:

   - Try with different special characters, such as colon, semicolon etc.
   - Try a test case where the second word deleted is not numeric.
   - Try with the cursor on the dash, or before the dash.
   - Try with the dash elsewhere in the word, such as the beginning.
   - Try with multiple dashes.
   - Try with tabs instead of spaces.
   - Try to replicate in a different operating system.
   - Try to replicate with a different Open Office version.

   Other correct answers are also possible.

2. **[20 marks]** Consider a logic function Z of four boolean variables A, B, C, D. The formula for Z is

$$Z = A + BC + CD + BD$$

Describe the process of applying the Variable Negation test strategy. What is the minimum test suite suggested by this strategy?

Following is the truth table (not necessary to draw for full marks).

Variant	A	B	C	D	Z
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

The following table presents all the candidate sets. Each member of a set was worth one mark. Each incorrect member was -1 mark.

Candidate set #	Term	Type	Set
1	`A`	Unique true point	{8,9,10,12}
2	`A`	Near false point `~A`	{0,1,2,4}
3	`BC`	Unique true point	{6}
4	`BC`	Near false point `~BC`	{2}
5	`BC`	Near false point `B~C`	{4}
6	`CD`	Unique true point	{3}
7	`CD`	Near false point `~CD`	{1}
8	`CD`	Near false point `C~D`	{2}
9	`BD`	Unique true point	{5}
10	`BD`	Near false point `~BD`	{1}
11	`BD`	Near false point `B~D`	{4}

Candidate sets of cardinality 1 determine that variants 1,2,3,4,5,6 must be part of the test suite. This covers all candidate sets except for the first one. We select one member from that set as well. The final test suite (worth 3 marks) will contain the following test cases:

1
2
3
4
5
6
8 or 9 or 10 or 12

3. **[20 marks]** Consider the following function

```
public void printSeq() {
     int lastDigit = seq[len];
     if (len == n)
     {
         if (isPrime(lastDigit + 1))
         {
             for (int i = 1; i <= n - 1; i++)
             {
                 System.out.print(seq[i] + " ");
             }
             System.out.println(seq[n]);
         }
     }
     else
     {
         for (int i = 2; i <= n; i++)
         {
             if (isPrime(lastDigit + i) && (notInSeq(i)))
             {
                 seq[len + 1] = i;
                 len++;
                 printSeq();
                 len--;
             }
         }
     }
 }
```

Draw the Control Flow Graph for this function. Clearly indicate what are the segments.

The 12 segments for printSeq (each worth 1 mark) are shown below. The Control Flow Graph is shown in the next page (8 marks). Each incorrect edge is minus one mark. Segment Z denotes the end of the execution of printSeq.

A

```
     int lastDigit = seq[len];
     if (len == n)
```

B

```
         if (isPrime(lastDigit + 1))
```

C

```
for (int i = 1;
```

D

```
i <= n - 1;
```

E

```
i++)
{
    System.out.print(seq[i] + " ");
}
```

F

```
System.out.println(seq[n]);
```

G

```
for (int i = 2;
```

H

```
i <= n;
```

I

```
i++)
```

J

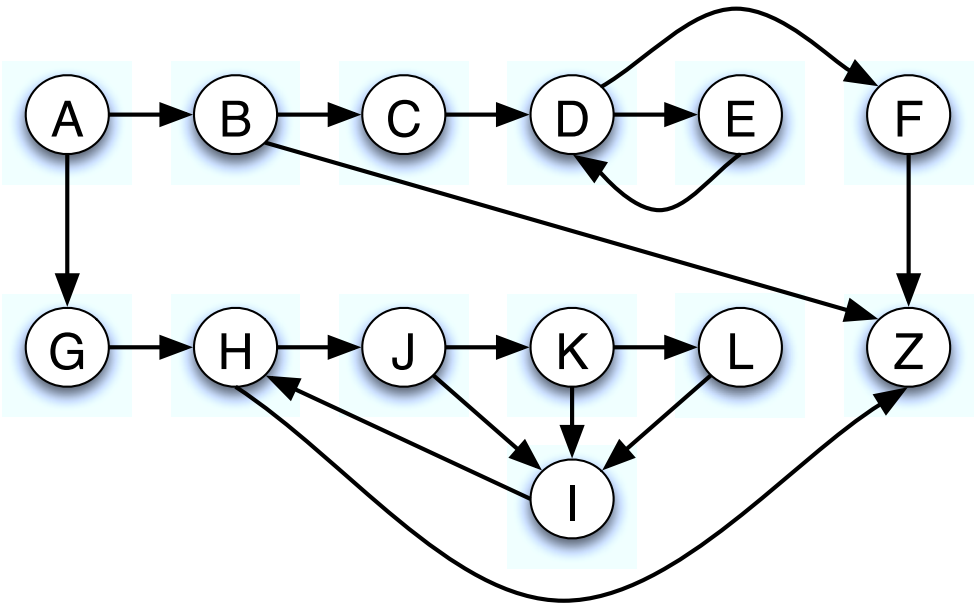
```
if (isPrime(lastDigit + i)
```

K

```
(notInSeq(i)))
```

L

```
seq[len + 1] = i;
len++;
printSeq();
len--;
```



4. **[20 marks]** Open Office Writer has an auto-format feature that recognizes ordinal numbers and automatically superscripts the ordinal ending. For example, if the user types the word `1st`, then it is automatically transformed to 1^{st}. This behaviour should occur only for appropriate ordinal endings of two characters, and only if the ordinal number is a separate word (words are separated by spaces or tabs).

   Your task is to describe the application of any software testing strategies that you believe are appropriate in order to test this feature. The derived test cases must be identified clearly in your answer. Answers that list test cases without describing how they were derived will receive a poor mark.

   The specification above is provided as is. You may have to make assumptions about system behaviour that is not mentioned in the specification. If this is the case, choose reasonable ones and state them explicitly.

   Equivalence class testing is the test strategy that applies best. Following are a list of equivalence classes required for full marks. The classes can be expressed in different ways, but they still have to lead to an equally thorough test suite.

   (a) Correct ordinal numbers. Necessary subclasses include:
      i. Numbers ending in 1 (but not 11), ordinal ending is “st”.
      ii. Numbers ending in 2 (but not 12), ordinal ending is “nd”.
      iii. Numbers ending in 3 (but not 13), ordinal ending is “rd”.
      iv. Numbers ending in 4 - 0, ordinal ending is “th”.
      v. Numbers ending in 11, 12, 13, ordinal ending is “th”.

   (b) Incorrect ordinal numbers. Necessary subclasses include:
      i. Incorrect ordinal ending, e.g. “1sr”.
      ii. Correct but non-matching ordinal ending, e.g. “2st”.
      iii. Correct ordinal numbers not surrounded by spaces or tabs, e.g. “3rdd”.

   (c) Your test suite also needs to include the special cases of “0th”, as well as negative numbers such as “-1st”. The expected response is ambiguous in the spec, but the test case needs to be there nevertheless.

   The equivalence classes can also be expressed across more than one dimension, such as the numeric part and the character part. This should lead to a larger test suite that should receive full marks since it will derive the test cases described above.

   While it is possible to view the behaviour of this system as state-based, it is very ineffective. A statechart that would lead to an equally thorough test suite would be very complicated. Partial marks were given to solutions using state-based testing only if the statechart would provide a strong set of test cases.

5. **[20 marks]** In this question, you will be deriving test cases for a piece of software that helps a marketing company decide what products to market to a specific client. There are three inputs to the program: Gender, City Dweller, and age group: A (young), B (middle-aged), C (older).

   The company markets four products: W, X, Y, and Z. Product W will appeal to female city dwellers. Product X will appeal to young females. Product Y will appeal to male middle-aged shoppers who do not live in cities. Product Z will appeal to all but older females.

   Your task is to describe the application of any software testing strategies that you believe are appropriate in order to test this program. The derived test cases must be identified clearly in your answer. Answers that list test cases without describing how they were derived will receive a poor mark.

The complex logic in this problem indicates that a decision table is the right approach. Here are all 12 combinations [10 marks]:

Rule #	1	2	3	4	5	6	7	8	9	10	11	12
Gender	F	M	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
W	X				X				X			
X	X		X									
Y								X				
Z	X	X	X	X	X	X	X	X		X		X

Rules 2,4,6,7,10,12 have the same action set, so it is possible to reduce the decision table. There are two ways to do the reduction, either one worth 10 marks:

(a) Combine rules 2,6,10 resulting in the following decision table (the numbering of rules is retained for demonstration purposes)

Rule #	1	2	3	4	5	7	8	9	11	12
Gender	F	M	F	M	F	F	M	F	F	M
City	Y	Y	N	N	Y	N	N	Y	N	N
Age	A	-	A	A	B	B	B	C	C	C
W	X				X			X		
X	X		X							
Y							X			
Z	X	X	X	X	X	X	X			X

(b) Combine rules 2,4 and rules 10,12 resulting in the following decision table (the numbering of rules is retained for demonstration purposes)

Rule #	1	2	3	5	6	7	8	9	10	11
Gender	F	M	F	F	M	F	M	F	M	F
City	Y	-	N	Y	Y	N	N	Y	-	N
Age	A	A	A	B	B	B	B	C	C	C
W	X			X				X		
X	X		X							
Y							X			
Z	X	X	X	X	X	X	X		X	