SUGGESTED ANSWERS AND COMMENTS
**QUESTION A.1:**

When testing a *Scribble* application, a unit test is a test of ...

E. an individual script or sprite.

**QUESTION A.2:**

When testing a *Scribble* application, an integration test is a test of ...

D. the interaction between sprites

**QUESTION A.3:**

A(n) ________ loop has no way of ending and repeats until the program is interrupted.

C. infinite

**QUESTION A.4:**

A __________ is a diagram that graphically depicts the steps that take place in a program.

C. flowchart

**QUESTION A.5:**

A(n) ________ is a set of well-defined logical steps that must be taken to perform a task.

D. algorithm

**QUESTION A.6:**

Consider the following *Scribble* script.

![Scribble script](image)

The output will be:

C. true
QUESTION A.7:

Consider the following Scribble script.

The output will be:

C. true

QUESTION A.8:

Consider the following Scribble script.

The output will be:

D. false

QUESTION A.9:

Read the following code segment. Assume that x is an integer variable that has been correctly defined and has been given a legal value.

Which option is true?

B. The "say" command will always execute.
QUESTION A.10:

What will happen once the following Scribble script has run?

```
set j \text{ to } 3
if j = 1
  set j \text{ to } 2
```

A. Result in $j$ having a value of 3.

QUESTION A.11:

Consider the following Scribble script.

```
set count \text{ to } 1
set x \text{ to } 0
repeat until not x < 4
  set j \text{ to } -3
  set x \text{ to } x + 1
repeat until not j > -5
  set j \text{ to } j - 2
  say Hello! for 2 secs
```

How many times will 'Hello!' be printed out?

A. 4.

QUESTION A.12:

When possible you should avoid using _________ variables in a program.

A. global
**QUESTION A.13:**

The following *Scribble* script is expected to display all numbers of the 8 times table (ie 8, 16, 24, 32, etc) less than 100.

What will actually happen?

E. It will display one number too many.

**QUESTION A.14:**

Consider the following *Scribble* script.

This code is supposed to ask the user to enter a value. The value entered is then stored in the variable named `value`. That number (value) squared is then stored in the variable named `squared`. The user is then shown the original number entered and its square. For example, if a user typed 3, the program should respond with "3 squared is 9". What will actually happen when this code is run?

B. The code will work, but will be stuck in an infinite loop.
QUESTION A.15:

In Scribble, a variable that is visible to every sprite in the program is a ________ . A. universal variable

C. global variable

QUESTION A.16:

In a nested loop, the inner loop goes through all of its iterations for every single iteration of the outer loop.

A. True

QUESTION A.17:

Consider the following Scribble script and select the correct option below.

A. Employee will be printed once to the screen and the loop will terminate.

The way Scribble works makes it possible for C to be a correct answer too. Forever loops operate a bit strangely in Scribble. This one stays active until all scripts are stopped, however, the code will run just once.

QUESTION A.18:

Consider the following Scribble script.

Which one of the following is true?

D. "Employee" will be printed on the screen when the code executes.
QUESTION A.19:

A _______ is a Boolean variable that signals when some condition exists in the program.

A. flag

QUESTION A.20:

Consider the following Scribble script. Assume that letter1, letter2 and letter3 are text variables that have been correctly defined and that letter1 has a value of "d", letter2 has the value "a", and letter3 has the value "m". After the script has run what will be the output if letter1, letter2 and letter3 are output (in that order)?

A. m, d, a
B.1 Debugging

You are working in your first job as a programmer. It's late in the afternoon and a you are aware of a colleague working near you who is becoming quite anxious and frustrated as they try to debug an application program that they have been working on. Your colleague is also a recent graduate – from another University. The code they are working on is largely finished but some parts of the application aren't working correctly. You decide to help your colleague. What advice will you give them about the strategies they should be using to find and fix bugs in their program code?

This question really is an invitation to tell us all you know about debugging. You can draw directly from the material on debugging in week 2 and week 12, and also the “general” approach to problem finding we adopted during labs and in the unit in general. There’s lots of scope for the answers here to be quiet different in their “shape” and flow. Aside from the specific points listed below, good answers will include examples and explanations to show that the material hasn’t just been rote learned. The main points to cover in the answer are:

• Debugging your own code can be hard, we see what we think its there rather what is actually there. Code reviews and walkthroughs … are among the techniques that can be used to get another set of eyes to examine code and find the errors that the coder can’t see.

• Desk checks (manual or automatic) and step by step running of the code (with an “industrial” IDE) can be useful to stop and fix errors. Esp. those related to the incorrect start and end of loops.

• There are plenty of techniques that can be used to work out where a program is failing - viewing variables, adding stop points, “say” statements, playing a sound etc. when a bit of code is running. (see lecture 2)

• Sometimes you have to take a break. Problems sometimes solve themselves when you leave them for a while.

• Ask for help. Don’t just sit staring at a screen.
B.2 Desk-checking code 5 marks

Perform a desk check that shows the step-by-step operation of the following Scribble script. Assume that the list named testScores has global scope and contains 12 numeric items (shown below).

<table>
<thead>
<tr>
<th>Step</th>
<th>n</th>
<th>scoreAverage</th>
<th>sum</th>
<th>length(TestScores)</th>
<th>item (n) of (testScores)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td>0</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
<td>18</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
<td>24</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
<td>36</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>0</td>
<td>42</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>0</td>
<td>54</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>0</td>
<td>60</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0</td>
<td>72</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0</td>
<td>78</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>90</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>0</td>
<td>96</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>0</td>
<td>108</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>9</td>
<td>108</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>
In Scribble, it is possible to create custom code blocks. These code blocks can be commands, reporters or operators. During the semester several custom code blocks were demonstrated. One was named the *DrunkardsWalk*. This code block was a command code block. When used, it was given two parameters that defined the behaviour of a sprite on the stage that drew a coloured random grid pattern. Another was a reporter block that converted degrees Celsius to degrees Fahrenheit. A custom operator code block was also created to perform the logical test \( \geq \) (greater than equals to).

Discuss the possible benefits of the use of custom code blocks. Make sure in your answer you discuss each of the examples cited above.

Rather like question B.1, this is also an invitation to tell us all you know about code structure. We spent quite a bit of time in lectures talking about the benefits of code blocks. You can draw directly from the material from lectures, and also the “general” approach to code structure we adopted during labs and in the unit in general. As for question B.1, there’s lots of scope for the answers here to be quiet different in their “shape” and flow. Aside from the specific points listed below, good answers will include examples and explanations to show that the material hasn’t just been rote learned. The main points to cover in the answer are:

- Code blocks encourage reuse. That saves time and effort. And reduces the chances of making mistakes.
- They make scripts shorter and as a result more readable.
- They make scripts more understandable (providing the code blocks) are given good names that reflect their function.
- Code blocks help with functional decomposition, the actual structure of the code can match the flowcharts and diagrams used to design the application. The development process can match that structure too.
A *Scribble* application asks the user to enter a distance in miles. It then converts the distance entered to kilometres and displays the result of that conversion to the user. Create a set of test data items that could be used to make sure that the program was operating correctly.

We spent a lot of time early in the unit working on the development of test data. Check week 2, and esp. the work you did in tutorial 2 (and the tutorial 2 test). The idea to to develop significantly different options ... not different versions of data that are similar. In answering this question ... look to include normal inputs, incorrect inputs and weird inputs. Look esp. for boundary conditions. The question doesn’t say it explicitly but I don’t think negative inputs should be valid. The answer will be a set of test data ... not a description of how they were developed.

Normal inputs (for example):

50, 100000, 9999999999999

Boundary and errors:

0, -50, 100000, -99999999999999, hello, 2hello, @#$%^, 3%^&6%

Weird inputs:

Might include control or esc characters, characters in different languages/fonts.
B.5  Creating a flowchart  5 marks

Draw a flowchart to represent the following Scribble script. The script implements a version of the Binary Search algorithm. The script runs by asking the user to enter the name of a golfer. It then searches for that name in a sorted list named MastersWinners. (The list contains the names of all the golfers who have won the U.S. Masters golf tournament.)
A Scribble sprite (named Search) contains an unsorted list of numbers (called numberList). Write a script that runs when that sprite is clicked. The script will ask the user to enter a number. The script will search for that number in numberList. If the number is found in the list it will display a message that says that the number was found and what the index number of the entry in the list is. If the number isn’t found the script will report that the number isn’t in the list.

This script uses:

- a variable named `number` - which stores the user entered number that will be searched for.
- a Boolean variable named `found` - which is set to true if the number searched for is found in the list.
- a variable named `index` - which is used a utility variable to keep track of the list item that is the subject of the check, the list items are checked one at a time as the search is conducted.

Note that script - should start when search clicked - not the green flag.
A **Scribble** sprite (named *Sprite1*) contains list named *Numbers*. Write a script that runs when the green flag is clicked. When the script runs it will first delete any data already in the list *Numbers*. The script will then fill the list *Numbers* with the numbers of the Fibonacci sequence (in order).

The Fibonacci sequence starts with the numbers 1, and 1, and then the next numbers in the list are the previous two added together. So the 3\(^{\text{rd}}\) item is 1+1 = 2, the 4\(^{\text{th}}\) item is 1+2 = 3, the 4\(^{\text{th}}\) item 2+3 = 5, and so on.

The script will begin by asking the user to enter a value. That value will be length of the list that is to be created. For example, if the user types in 6, the script will then create a list with 6 values of the Fibonacci sequence (1, 1, 2, 3, 5, 8).

This script uses:
- a Boolean variable named *elements* - which stored the user entered number representing the number of elements to include in the sequence generated.
- a variable named *previous* - which holds the second of the two numbers used to create the next value in the sequence
- a variable named *current* - which holds the current value in the sequence
- a variable named *next*, used to calculate the next value (by adding *current* and *previous* together)
- a variable called *count* - which is used a utility variable to keep track of the number of iterations that have been performed as the sequence is created
A *Scribble* application contains a list of names, named *Names*. The length of the list *Names* changes as the application runs. The list *Names* has been created with global scope. A sprite named *Sort* is visible on the stage. Write a script that runs when the sprite *Sort* is clicked. When the script runs it should use the BubbleSort algorithm to sort the items in the list *Names* alphabetically from A to Z.

The BubbleSort algorithm works, starting with the first pair, by comparing each pair of items in a list. If they are out of order they are swapped. It then examines the second pair of numbers in the list, and if they are out of order, they are swapped. This continues until the end of the list. This is repeated until the list is sorted. Make the script as efficient as the BubbleSort method can be made.

This script uses:

- a Boolean variable named *sorted* - which is set to true if the list is sorted and false if a swap is made.
- an integer variable named *index* - this is a utility variable used to traverse the list and access elements.
- a variable named *tmp* - used to temporarily hold a value in the list while a swap is performed.
Write a Scribble application scripts that lets a user play the game Tic-Tac-Toe. The game should be played by two players, taking turns to mark a 3x3 game grid with a X or a O. The first player to get three of their marks in a horizontal, vertical or diagonal row wins. If the game grid is filled with marks and no player has won, the game is drawn.

Answer the question by listing the sprites required. Also, describe the variables needed. This description should include the name of each variable, its purpose, and its scope. All the assets required to implement the game (graphic images and sounds) should be sketched or described. All of the scripts required should be written.

The answer to this can be taken directly from the lectures. There were two versions presented in lectures - in weeks 9 and 10.

In week 9 look at the slides numbers 14 to 43. The final app developed in week 9 is a great “model” for the answer to this question. It can be downloaded from Moodle. The lecture recording that covers the development of this application is located here http://vishnu.infotech.monash.edu.au/pod_files/Lecture%209%20CA.mp4. The discussion starts at the 19 minute mark (exactly).

In week 10 look at the slides numbered 20 to 23. The refactored version of the app developed in week 10 is another great “model” for the answer to this question. It can also be downloaded from Moodle. The discussion of the development of this version of the application is located here: http://vishnu.infotech.monash.edu.au/pod_files/Lecture%2010%20CA.mp4. The discussion starts at the 38 minute, 40 second mark.

Either of the two approaches ... not using clones, and using clones is fine ... the main thing is to “write” an application that works. If this was an essay question, you would expect that you would plan your answer before writing it ... the same applied to a coding question (this includes the questions in part C, but more so for this one as it's harder). There aren’t marks given explicitly for the “plan” - which would take the form of design models like a flowchart and/or a storyboard - but its hard to write a complex piece of software without doing that kind of planning. If you didn’t complete the question - perhaps you run out of time - or get some of the code detail wrong, then we can look at the “plan” you made an award marks for that ... just as we would for an unfinished or incomplete essay answer. So while the question doesn’t say draw a flowchart, or storyboard, or do any other kind of design or planning it’s hard to see how you could answer the question without doing that kind of work.