University of Stirling Computing Science and Mathematics

CSCU9A1 Tutorial 3 Solutions

1. Consider the following fragments of code:

```
sum = 0 ;
if (xvalue > 0)
{
  sum++;
}
if (yvalue > 0)
{
  sum++;
}
```

and

```
sum = 0 ;
if (xvalue > 0)
{
   sum++;
}
else if (yvalue > 0)
     {
        sum++;
     }
```

and

```
sum = 0 ;
if (xvalue > 0)
{
     sum++ ;
     if (yvalue > 0)
     {
        sum++ ;
     }
}
```

What is the difference in their effect? For each code fragment, draw up a *truth table* for the conditions `xvalue > 0` and `yvalue > 0`, showing the final value of `sum` in each case.

First one: sum may end up as 0, if xvalue <= 0 AND yvalue <= 0, as 1 if either xvalue <= 0 or yvalue <= 0 but not both, or 2 if both xvalue > 0 AND yvalue > 0. Both if statements always get

executed.

Second one: if xvalue > 0 sum will be incremented, and if xvalue < = 0 and yvalue > 0 sum will be incremented. Sum will never be incremented twice, since only one branch of the if statement will be executed.

Third one: if xvalue <= 0 sum is not incremented. If xvalue > 0 sum is incremented, and if yvalue is also > 0 sum is incremented again.

Truth tables. You have seen the standard truth tables for AND, OR, and NOT in lectures. Strictly speaking, the tables below should not be called truth tables, because the entry in the third column is an integer rather than a boolean. But this is a very useful kind of table to use when designing and testing complicated case-based logic, and it is a similar idea to a truth table. (Maybe we could call it a case table instead?)

	xvalue > 0	*yvalue > 0*	*sum*
1	*false*	*false*	*0*
	false	*true*	*1*
	true	*false*	*1*
	true	*true*	*2*
2	*false*	*false*	*0*
	false	*true*	*1*
	true	*false*	*1*
	true	*true*	*1*
3	*false*	*false*	*0*
	false	*true*	*0*
	true	*false*	*1*
	true	*true*	*2*

2. Consider the following fragment of code

```
String gradeLetter = "F" ;
if (grade >= 90) gradeLetter = "A" ;
if (grade >= 80) gradeLetter = "B" ;
if (grade >= 50) gradeLetter = "C" ;
```

What does the above code do, and what do you think it was intended to do? How would you modify it to make it do what was intended?

What does it do: If grade >= 50, gradeLetter is set to "C", and if grade < 50 gradeLetter is set to F. The middle lines actually have no effect at all, which is almost certainly unintentional (otherwise why write them?)
What's intended is that if the grade is >= 90, gradeLetter should be A, etc. How to get this

effect? Either use nested if statements, or simply put the last 3 lines in the opposite order (!). Better to use nested if statements…

3. Write a loop that computes and prints all the squares of positive integers less than *n*. Now write a program that computes all the integers whose squares are less than *n*.

```
int i = 0;
while (i < n)
{
      System.out.println(Math.pow(i,2));
      i++;
}

int i = 0;
while (Math.pow(i,2) < n)
{
      System.out.println(i);
      i++;
}
```

4. Write a loop that computes the sum of all the even numbers between 2 and 100, inclusive. Now modify the program to compute the sum of all the even numbers between 46 and 1322554 inclusive.

```
int sum = 0 ;
for (int i = 0; i <= 100 ; i = i + 2)
{
      sum = sum + i ;
}
System.out.println("sum of the numbers  = " + sum) ;
```

A variety of solutions are possible. e.g.

```
int startPoint = 46 ;
int endPoint = 1322554 ;
int sum = 0 ;
for (int i = startPoint; i <= endPoint; i = i + 2)
{
      sum = sum + i;
}
```

5. What is an infinite loop (non-hint: what is the address of the Apple company in Cupertina, California). How do you terminate a program that is in an infinite loop (ii) on your laptop or desktop and (ii) on your smartphone?

Apple's address is 1 Infinite Loop, Cupertino, California…

```
// This program contains an infinite loop
public class InfiniteLoop
{
    public static void main (String[] args)
    {
        while (true);
    }
}
```

If it is possible to write a shorter infinite loop in Java, I don't know how.

If a program running in BlueJ gets stuck in an infinite loop, it can be terminated by resetting the Java virtual machine (right click on the striped "barber's pole" icon at the bottom left of the main BlueJ window).

If the program is running in a terminal window, control-C may work. Otherwise, you probably need to go into (on a Mac) Force Quit, and select it and quit it manually, or under Windows, using the task manager.. And on a smartphone? I think you probably have to restart it!

6. Discuss the difference between the loop types supported in Java.

Covered while do, do while, for. While..do, may never execute, do..while always executes at least once, for when you know how often you want to go round the loop. Note that While..do can be used for all of these…

When to use each type?

`while` *loop. Use this when you know under what conditions the loop body should continue to be executed.*

`for` *loop. Use this if you know exactly how many times the loop body should be executed.*

`do-while` *loop. Use this if you know that the loop must always be executed at least once, and you also know under what conditions the loop body should continue to be executed.*