Prelim 1 Grading Guide/Solutions

Problem 1, Part A (10 points total)
The correct answer is:

The value of $x$ is ______26_____. (3 points)
-1 for $x = 28$.
-2 for $x = 4$.
-2 if the $x$ value is not an integer.

The value of $y$ is ______16_____. (3 points)
-1 for $y = 15$ or $y = 32$.
-2 for $y = 30$.
-2 if the $y$ value is not an integer.

The value of $b$ is ___false_____. (2 points)
-1 if used 1 for true and 0 for false.
-2 for 'true'.
-2 if the $b$ value is not a boolean.

The value of $c$ is ___true_____. (2 points)
-1 if used 1 for true and 0 for false.
-2 for 'false'.
-2 if the $c$ value is not a boolean.

Problem 1, Part B (10 points total)
Each blank is worth 1 point. The correct answer is:

\[
\begin{array}{c|c|c}
\text{x} & \text{y} \\
\hline
\text{int } x = 10; & 10 & \text{int } y = 5; & 5 \\
\text{y = x++;} & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\text{if (y <= x)} & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\{ & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\text{y = 2 - y * 2;} & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\} & \text{else} & \text{blank} & \text{blank} \\
\{ & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\text{y = 2 * x - 2;} & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\} & \text{x = -x + 10 \% 5;} & \_11\_ & \_10\_ & \_11\_ & \_10\_ \\
\end{array}
\]
Problem 2 (12 points total: 6 correctness, 6 style)

// Determine which integer is the smallest, and
// assign “smallest” to have the smallest integer’s value.

Here is one solution:

```c
if (n1 < n2)
{
  if (n1 < n3)
  {
    smallest = n1; // n1 < n2 && n1 < n3
  }
  else
  {
    smallest = n3; // n1 < n2 && n3 < n1
  }
}
else
{
  if (n2 < n3)
  {
    smallest = n2; // n2 < n1 && n2 < n3
  }
  else
  {
    smallest = n3; // n2 < n1 && n3 < n2
  }
}
```

Correctness (6 total):
2 points each for correctly determining when each number is the smallest (n1, n2, n3)
• -2 for backwards assignments (e.g., n2 = smallest; )

Style (6 total):
1 point: use of curly braces
1 point: indentation
2 points: each assignment should be commented (0.5 each)
2 points: using ‘else’ instead of ‘else if (…)’ for both final cases

NOTE: We did not take off points for missing semicolons or braces, but in the future we will be very strict about this.
Here is another solution:

```cpp
if (n1 < n2 && n1 < n3)
{
    smallest = n1;
}
else if (n2 < n1 && n2 < n3)
{
    smallest = n2;
}
else    // must be the case that n3 < n1 && n3 < n2
{
    smallest = n3;
}
```

Correctness (6 total):
2 points each for correctly determining when each number is the smallest (n1, n2, n3)

Style (6 total):
1 point: use of curly braces
1 point: indentation
2 points: commenting the final else statement
2 points: using ‘else’ instead of ‘else if (…)’ for the final case
Problem 3, Part A (4 points):

For loops and while loops can both be used to repeat a section of code. If it is not known ahead of time how many times the loop should execute, a while loop is appropriate. If the start and end points are well-defined (and therefore, the number of times the loop should execute are known, or computable), then a for loop is appropriate.

Grading:
2 points for indicating when a while loop should be used.
2 points for indicating when a for loop should be used.
-2 if they compare a while loop to a do-while loop.

Problem 3, Part B (8 points):

2 points for each of the following (1 point for name, 1 point for description):

1. Loop setup: initialize any loop variables
2. Loop condition: how to check for termination (when the loop should exit)
3. Loop body (processing): the actual work done by the loop during each iteration
4. Loop update: update loop variables appropriately (usually variables that show up in the condition)
   - -2 for including “loop post-processing”
   - -0.5 for each step that is specific to one kind of loop only (e.g. only mentions for loops)
Problem 3, Part C (6 points: 3 correctness, 3 style):

Here is the solution:

```java
// ------ Translated while loop ---------
int f = 0; // initialize the loop variable
// Loop the specified number of times
while (f < numFlips) // check the condition
{
    // This generates a random integer, either 0 or 1.
    // Treat 1 as heads, 0 as tails
    int flip = (int) (Math.random()*2);
    // Update your statistics here:
    if (flip == 1)
    {
        numHeads++;
    }
    else
    {
        numTails++;
    }
    f++; // Move the update here
}
```

Correctness:
1 point for each added line (initialization, condition, update)
- -1 for f = 0; instead of int f = 0;
- -1 for incrementing numFlips instead of f
- -1 for f <= numFlips instead of f < numFlips

Style:
1 point for each added line (each should have a comment)
- -1 if f++; is not the last statement in the loop
- -1 for f = f++;
Problem 4 (20 points):

Here is one solution:

1. **Input**: a text document consisting of zero or more words
2. **Process**:
   a. Initialize two variables:
      i. numWords to 0 (total number of words in the document)
      ii. numWrong to 0 (total number of misspelled words)
   b. Loop through the words in the document. For each word,
      i. Increment numWords
      ii. If the word is capitalized, skip to the next word
      iii. Otherwise, check if the word is in the dictionary. If so, go to the next word
      iv. Otherwise, increment numWrong and go to the next word
   c. Calculate the error rate:
      i. If numWords is 0, set errorRate to 0.0
      ii. Otherwise, set errorRate to numWrong / numWords, remembering to cast numWords to a double
3. **Output**: the error rate: errorRate * 100 (to get a percentage)

We did not penalize for forgetting to check if numWords is 0. Division by zero is a serious problem, and in any real program you must check for conditions such as these.

We also did not penalize for solutions which only incremented numWords for non-capitalized words. The specification could reasonably have been interpreted either way.

Here’s how the points were assigned:

- **5 points for Input**
  - 1 point: including the word “Input”
  - 2 points: having some kind of input
  - 2 points: indicating that the input is a document
  - -1 for specifying that the input is a word

- **5 points for Output**
  - 1 point: including the word “Output”
  - 2 points: outputting the error rate
  - -1 for outputting number of errors instead of error rate (it’s ok if both are given)
  - -1 for outputting a list of misspelled words, or other output not called for
  - 2 points: indicate how error rate is calculated
  - +1 for checking whether number of words is 0, and indicating what to do if it is 0

- **5 points for Process correctness**
  - 1 point: including the word “Process”
  - 1 point: correct update of number of missspelled words
  - 1 point: correct update of number of total words
  - 1 point: some notion of a loop or iterating over every word in the document
  - -1 for infinite loops
  - 1 point: skipping capitalized words

- **5 points for Process style**
  - 1 point: check for a capitalized word before checking whether it’s misspelled
  - 4 points: proper level of detail (take points off as necessary)
-1 for incrementing number of errors when the word is “spelled incorrectly” (should indicate to check this against the dictionary)
-1 for counting the number of words before/after looping through to check for misspelled words (efficiency)
-1 for not numbering steps
-1 for unnecessarily repeating calculations
-1 for excessively vague steps
-1 for misreading directions about assumptions you can make
-1 for ignoring case of words: should skip capitalized words instead
-1 for extra output
-1 for not initializing counter variables