CSCE 110: Programming I

Exam #1 — Answer Key

March 5, 2012

Your final score is circled in black. There are three values on your exam. The first, second, and third numbers represent the number of points you received for questions 1 through 7, question 8, and completing the survey (bonus points), respectively.

Below are the solutions for Versions Gig ’Em and Reveille of Exam #1 along with the regrading policy.

1 Regrading Policy

If there are any grading errors related to your exam, you must notify me in writing by March 8, 2012. After March 8th, no changes to exam grades will be considered. Below, are the steps that you must follow if you want your exam regraded.

1. Write a formal statement that specifies clearly the error in question.

2. Attach your statement to your exam.

3. During class or office hours, give me your statement along with your exam to reconsider.

Finally, if your grading error is related to wanting to receive more partial credit (on question 8 for example), then your exam will be returned back to you. However, if there is an actual error (e.g., a correct solution is marked incorrect, your exam score is not tallied correctly), then please follow the above steps to have your exam regraded.

2 Exam Solutions

2.1 Version Gig ’Em

1.
   a) False
   b) True
   c) True
   d) True
   e) False
   f) False
2.

a) 1, 2, 3, 4, 5, 7, 8, 9
b) p
c) rrrr
d) pear
e) raep
f) e
g) pearXapple
h) True

3.

a) int, while, if, elif, else, print (any 3 of these would be correct answers)
b) 1, 2, 3, 4, 6, 7, 10, 3, 12, 13
c) result is 4
d) int_value is 3
e) result is 0

4.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value assigned to the variable</th>
<th>Data type of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>['apple', ['pear', 13, -14.2], ('yes', 'no')]</td>
<td>list</td>
</tr>
<tr>
<td>b</td>
<td>('yes', 'no')</td>
<td>tuple</td>
</tr>
<tr>
<td>c</td>
<td>13</td>
<td>int</td>
</tr>
<tr>
<td>d</td>
<td>'no'</td>
<td>string</td>
</tr>
<tr>
<td>e</td>
<td>('yes', 'no', 'yes', 'no')</td>
<td>tuple</td>
</tr>
<tr>
<td>f</td>
<td>['apple', ['pear', 13, -14.2]]</td>
<td>list</td>
</tr>
<tr>
<td>g</td>
<td>('yes', 'no')</td>
<td>tuple</td>
</tr>
</tbody>
</table>

5.

<table>
<thead>
<tr>
<th>Range of numbers</th>
<th>Maximum number of guesses using our optimal strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 10</td>
<td>4</td>
</tr>
<tr>
<td>1 to 100</td>
<td>7</td>
</tr>
<tr>
<td>1 to 1000</td>
<td>10</td>
</tr>
<tr>
<td>25 to 50</td>
<td>5</td>
</tr>
</tbody>
</table>

6.

a) int

b) 3 6 9
   4 8 12
   5 10 15

<table>
<thead>
<tr>
<th>3 6 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 8 12</td>
</tr>
<tr>
<td>5 10 15</td>
</tr>
</tbody>
</table>

c) 3 9 27
   4 16 64
   5 25 125

<table>
<thead>
<tr>
<th>3 9 27</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 16 64</td>
</tr>
<tr>
<td>5 25 125</td>
</tr>
</tbody>
</table>
7.

a) 1, 2, 3, 4, 5, 6, 7, 8, 5, 6, 8, 5, 6, 7, 8, 5, 9, 10
b) 
   | i: 3 |
   | j: 2 |
c) 
   | i: 3 |
   | j: 0 |

8.

The most popular approach the class used to solve this problem required two while loops. There were a handful of students that used for loops. I provide two examples of using for loops for you to study. Notice that the amount of code required is only a few lines.

Listing 1: q8-ver1.py

```python
symbol = raw_input('Symbol: ')
size = int(raw_input('Size: '))
print # print an empty line
for i in range(size):
    print symbol * (size - i)
for i in range(2, size + 1):
    print symbol * i
```

Listing 2: q8-ver2.py

```python
symbol = raw_input('Symbol: ')
size = int(raw_input('Size: '))
print # print an empty line
for i in range(size, 0, -1):
    print symbol * i
for i in range(2, size + 1):
    print symbol * i
```

2.2 Version Reveille

1.

a) False  
b) True  
c) True  
d) False  
e) False  
f) False
2. 
   a) 1, 2, 3, 4, 5, 7, 8, 9  
   b) a  
   c) llll  
   d) appl  
   e) elppa  
   f) r  
   g) True  
   h) appleXpear  

3. 
   a) int, while, if, elif, else, print (any 3 of these would be correct answers)  
   b) 1, 2, 3, 4, 6, 7, 10, 3, 12, 13  
   c) result is 2  
   d) int_value is 4  
   e) result is 1  

4. 

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value assigned to the variable</th>
<th>Data type of variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>['pear', ('no', 'yes'), ['orange', 31, -15.1]]</td>
<td>list</td>
</tr>
<tr>
<td>b</td>
<td>['orange', 31, -15.1]</td>
<td>list</td>
</tr>
<tr>
<td>c</td>
<td>'yes'</td>
<td>string</td>
</tr>
<tr>
<td>d</td>
<td>-15.1</td>
<td>float</td>
</tr>
<tr>
<td>e</td>
<td>['orange', 31, -15.1, 'orange', 31, -15.1]</td>
<td>list</td>
</tr>
<tr>
<td>f</td>
<td>['pear', ('no', 'yes')]</td>
<td>list</td>
</tr>
<tr>
<td>g</td>
<td>['orange', 31, -15.1]</td>
<td>list</td>
</tr>
</tbody>
</table>

5. 

<table>
<thead>
<tr>
<th>Range of numbers</th>
<th>Maximum number of guesses using our optimal strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 10</td>
<td>4</td>
</tr>
<tr>
<td>1 to 100</td>
<td>7</td>
</tr>
<tr>
<td>1 to 1000</td>
<td>10</td>
</tr>
<tr>
<td>25 to 50</td>
<td>5</td>
</tr>
</tbody>
</table>

6. 

   a) int  
   b)  
       2 4 6  
       3 6 9  
       4 8 12  
   c)  
       2 4 8  
       3 9 27  
       4 16 64
7.

a) 1, 2, 3, 4, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 8, 5, 9, 10

b) 

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

c) 

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

8. The most popular approach the class used to solve this problem required two while loops. There were a handful of students that used for loops. I provide two examples of using for loops for you to study. Notice that the amount of code required is only a few lines.

Listing 3: q8-ver1.py

```python
1 symbol = raw_input('Symbol: ')  
2 size = int(raw_input('Size: '))  
3  
4 print '# print an empty line'  
5 for i in range(size):  
6    print symbol * (size - i)  
7  
8 for i in range(2, size + 1):  
9    print symbol * i
```

Listing 4: q8-ver2.py

```python
1 symbol = raw_input('Symbol: ')  
2 size = int(raw_input('Size: '))  
3  
4 print '# print an empty line'  
5 for i in range(size, 0, -1):  
6    print symbol * i  
7  
8 for i in range(2, size + 1):  
9    print symbol * i
```