Sample Exam Questions
Section A

1. Consider the following Python script.

```
01  sentence = 'Beautiful is better than ugly'.split(' ')
02
```

What should be inserted at line 2 onwards to produce the following output:

```
['Beautiful']
['better', 'than']
ylgu
```

A.
```
print(sentence[0])
print(sentence[2:3])
print(sentence[4])
```

B.
```
print(sentence[0:1])
print(sentence[2:4])
print(sentence[4:5])
```

C.
```
print(sentence[1])
print(sentence[3:4])
print(sentence[5][::-1])
```

D.
```
print([sentence[0],sentence[1]])
print(sentence[2:4])
print([sentence[4][::-1]])
```

E.
```
print(sentence[0:1])
print(sentence[2:4])
print(sentence[4][::-1])
```

2. Consider the following program. What is the output when you run it?

```
x = { 'apple':1, 'orange':2, 'durian':3 }
x['apple'] = 4
x['coconut'] = 6
print(x)
```

A. `{'apple': 1, 'orange': 2, 'durian': 3}`

B. `{'apple': 4, 'orange': 2, 'durian': 3}`

C. `{'apple': 1, 'orange': 2, 'durian': 3, 'coconut': 6}`

D. `{'apple': 4, 'orange': 2, 'durian': 3, 'coconut': 6}`

E. script execution error

3. Consider the following Python code.
   Which range of value will cause "Yes" to be displayed?

```
if value < 70:
    if value < 30:
        print("No")
    elif value < 60:
        print("Yes")
```

☐ A. `value < 60`

☐ B. `30 <= value < 70`

☐ C. `30 < value < 70`

☐ D. `value >= 30`

☐ E. `30 <= value < 60`

4. Consider the following for statement. How many times will the loop be executed?

```
for i in range(1, 5):
    print("apple")
```

☐ A. 5

☐ B. 4

☐ C. 3

☐ D. 2

☐ E. 1

5. Consider the following program. What is the output when you run it?

```
flag = False

x = 2
while x <= 6:
    x += 1
    if x % 2 == 0 and x % 3 == 0:
        flag = not flag
    else:
        print (x)

print(flag)
```

A.
```
3
4
5
7
True
```

B.
```
3
4
5
6
7
True
```

C.
```
3
4
5
7
False
```

D.
```
2
3
4
7
True
```

E. Script execution error.

6. Consider the following code. What is the output when you run it?

```
a = 1
while 17 % a != 5:
    print(str(a)+ " " + str(17 % a))
    a += 1
```

☐ A.
```
1  1
2  0
3  0
4  1
5  1
```

☐ B.
```
1  0
2  1
3  2
4  1
5  2
```

☐ C.
```
1  0
2  1
3  2
4  3
5  4
```

☐ D.
```
1  1
2  0
3  0
4  1
5  1
```

☐ E.
```
1  1
2  2
3  1
4  2
5  1
```

7. What is the output when you run the following program?

```
y = 3
x = 2
x = 1
y = x

if x > 1:
    print("apple")
elif x < 4:
    print("orange")
else:
    print("grape")

print(x + 1)
print(y)
```

☐ A.
```
apple
3
1
```

☐ B.
```
apple
2
2
```

☐ C.
```
orange
2
1
```

☐ D.
```
orange
2
3
```

☐ E.
```
grape
2
1
```

8. What is the output when you run the following program?

```
x = [[1],[2,3],[4,5,6]]

print(len(x))
print(len(x[0]))

for i in range(len(x)):
    sum = 0
    for j in range(len(x[i])):
        sum += x[i][j]
    print(sum)
```

☐ A.
```
6
1
1
5
15
```

☐ B.
```
6
1
1
6
21
```

☐ C.
```
3
1
1
5
15
```

☐ D.
```
3
1
1
6
21
```

☐ E. Script execution error

9. Consider the following Python code. What is the output?

```
y = 8
x = 2 % 5
print(x / y)
```

☐ A. `0`

☐ B. `0.25`

☐ C. `2/8`

☐ D. `0.375`

☐ E. script execution error

10. Examine the following code segment, and then select the **CORRECT** output.

```
def test_one(x):
    x -= 1
    return x > 4

def test_two(x):
    print (x)
    return x <= 8

x = 5

if test_one(x) and test_two(x):
    print("apple")
elif x >= 5:
    print("orange")
```

☐ A.
```
orange
```

☐ B.
```
4
apple
```

☐ C.
```
4
orange
```

☐ D.
```
5
orange
```

☐ E. script execution error

11. Given the following code:

```
def perform_magic(first, second):
    third = first[0]
    first[0][0] = 5

    second.append(third)
    second[1][1] = 8
    second.append([6,7])

first = [[3,4]]
second = [[1,2]]

perform_magic(first, second)

print(first[0])
print(second[2])
```

What is the output when you run the following program?

☐ A.
```
[3,4]
[1,2]
```

☐ B.
```
[3,8]
[5,8]
```

☐ C.
```
[3,4]
[6,7]
```

☐ D.
```
[5,8]
[6,7]
```

☐ E. script execution error

12. What is the output when you run the following program?

```
def caps_display(fruit1, fruit2, fruit3):
    print(fruit1.upper(), fruit2.upper(), fruit3.upper())

def display(fruit1, fruit2,  fruit3):
    print(fruit1 + " " + fruit2 + " " + fruit3)
    fruit1 = fruit2
    fruit2 = "papaya"
    caps_display(fruit1, fruit2, fruit3)
    print(fruit1 + " " + fruit2 + " " + fruit3)

fruit1 = "apple"
fruit2 = "orange"
fruit3 = "pear"

# Note the order of the variables
display(fruit3, fruit2, fruit1)
caps_display(fruit1, fruit2, fruit3)

```

☐ A.
```
pear orange apple
ORANGE PAPAYA APPLE
orange papaya apple
APPLE ORANGE PEAR
```

☐ B.
```
pear orange apple
ORANGE PAPAYA APPLE
orange papaya apple
ORANGE PAPAYA APPLE
```

☐ C.
```
pear orange apple
PEAR ORANGE APPLE
pear orange apple
APPLE ORANGE PEAR
```

☐ D.
```
orange apple pear
ORANGE PAPAYA APPLE
orange papaya apple
PEAR ORANGE APPLE
```

☐ E.
```
orange apple pear
PEAR ORANGE APPLE
orange papaya apple
APPLE ORANGE PEAR
```

13. A method named hopscotch that accepts an integer parameter for a number of "hops" and prints a hopscotch board of that many hops. A "hop" is defined as the split into two numbers and then back together again into one. For example, hopscotch(1) should print:

```
***1***
2*****3
***4***
```

hopscotch(4) should print:

```
***1***
2*****3
***4***
5*****6
***7***
8*****9
***10***
11*****12
***13***
```

Which of the implementation(s) of hopscotch is/are correct?

I.
```
def hopscotch(x):
    count = 0
    i = 0
    while count < x * 2 + 1:
        i += 1
        if count % 2 == 0:
            print('***' + str(i) + '***')
        else:
            print(str(i) + '*****', end='')
            i += 1
            print(str(i))
        count += 1
```

II.
```
def hopscotch(hops):
    print('***1***');
    for i in range (1, hops):
        print(str(3 * i - 1) + '*****' + str(3 * i))
        print('***' + str(3 * i + 1) + '***')
```

III.
```
def hopscotch(hops):
    num = 1
    print('***' + str(num) + '***')
    num += 1
    for i in range (1, hops):
        print(str(num) + '*****' + str(num + 1))
        print('***' + str(num + 2) + '***')
        num += 3
```

☐ A. I only

☐ B. II only

☐ C. I and II only

☐ D. II and III only

☐ E. All of the above

Section B

Question 1

A. Write the `extract_unique_words` function in the file `question.py`. This method returns all the unique words in the list in the order of their first appearance in the `fruits` parameter.

The tester script is provided as shown below:

```
import question

fruits = ['apple', 'orange', 'pear', 'apple',
          'durian', 'apple', 'banana']

answers = question.extract_unique_words(fruits)
print(answers)
```

The output is then:

```
[apple, orange, pear, durian, banana]
```

```
// question.py
// Write your extract_unique_words method here
def extract_unique_words (fruits):
```

Question 2

Write a Python script, **adder.py** that continuously does the following:

- Prompts the user for an integer number
- Adds the number to the previous numbers read so far
- If the sum of all the numbers read exceeds 30, the program prints out the sum and quits the application.
- Otherwise, the program prints the message "The sum has yet to exceed 30".

The following is a sample run of the program:

```
C:\is111> python adder.py
Enter a number >12
The sum has yet to exceed 30.

Enter a number >18
The sum has yet to exceed 30.

Enter a number >1
The sum is 31!
```

Question 3:
Write a function called `convert_date` that takes in a date of type `str` in this format `"13-Nov-2013"` and returns another date of type `str` of this format `"Nov 13 2013"`. Note that the date may be a single digit – i.e. the input String is possibly `"9-Dec-2013"`.

Question 4:

A. Do a code walkthrough (trace) of the following program. List the output.

```python
i = 18
while i > 0:
    if i % 2 == 0:
        print(i - 1)
        i -= 3
    elif i % 3 == 0:
        print(i * i)
        i -= 1
    else:
        i += 1
```

Output

Question 5:

A cheerful number is a positive integer that is divisible by the sum of its odd digits. An example is the integer 156. The odd digits in 156 are 1 and 5, the sum of which is 6. Since 156 is divisible by 6 with no remainder, 156 is a cheerful number.

If the sum of a number's odd digits is 0, it is not a cheerful number (e.g. 20)

Write a program that prompts the user for a number between 0(exclusive) and 200(inclusive) and prints whether the integer is cheerful. The program keeps prompting until the user enters a valid number. You can assume that the user will always enter a numeric value.

A sample output is shown below:

```
D:\is111-exam> python checker.py
Enter num >-1
-1 is invalid.

Enter num >322
322 is invalid

Enter num >16
16 is cheerful.

D:\is111-exam> python checker.py
Enter num >154
154 is not cheerful.
```

Answer:

Question 6

This program prompts the user for three string values, and informs the user whether he has entered 1, 2 or 3 unique strings. Complete the `num_unique` function.

A sample output is shown below:

```
D:\is111-exam> python unique.py
Enter string 1 >apple
Enter string 2 >orange
Enter string 3 >pear
3 unique

D:\is111-exam> python unique.py
Enter string 1 >apple
Enter string 2 >apple
Enter string 3 >pear
2 unique

D:\is111-exam> python unique.py
Enter string 1 >apple
Enter string 2 >apple
Enter string 3 >apple
1 unique

D:\is111-exam>
```

Answer:

```
// write the num_unique function here













# prompt for 3 numbers
value1 = input ('Enter string 1 >')
value2 = input ('Enter string 2 >')
value3 = input ('Enter string 3 >')

result = num_unique(value1, value2, value3)
print (result, 'unique')
```

Question 7

The guessing game is a game where the computer thinks of a random number between 0 and 100 but keeps it secret from the player. The player will try to guess the number picked by the computer. If the player guesses correctly, the program will report the number of guesses that the player made.

Below is a sample run:

```
C:\is111-exam> python guessing_name.py

Welcome to Guessing Game!

Your guess (0 - 100)? 30
Your guess is too low
Your guess (0 - 100)? 40
Your guess is too low
Your guess (0 - 100)? 50
Your guess is too low
Your guess (0 - 100)? 60
Your guess is too low
Your guess (0 - 100)? 70
Your guess is too low
Your guess (0 - 100)? 80
Your guess is too low
Your guess (0 - 100)? 90
Your guess is too low
Your guess (0 - 100)? 91
Your guess is too low
Your guess (0 - 100)? 92
Bingo!
You got it right in 9 tries.
Another game (Y or N)? Y

Your guess (0 - 100)? 50
Your guess is too high
Your guess (0 - 100)? 40
Your guess is too high
Your guess (0 - 100)? 30
Your guess is too high
Your guess (0 - 100)? 14
Bingo!
You got it right in 4 tries.
Another game (Y or N)? N
Your average guess per game is 6.5
```

You have been given a buggy implementation of the program. Identify and correct **ALL** compilation and logic errors (i.e. errors that cause the program to behave incorrectly when executed) in *guessing_game*.

Note: 1 mark will be deducted for every incorrect error that you have identified.

```
# A robust number-guessing game with hinting.
import random

total_guess = 0
num_games = 0
num_guesses = 0

print("Welcome to Guessing Game!")

while reply == "Y":
    # pick a random number from 0 to 100 inclusive
    # there is no error in the line below
    # equivalent to
    # number = random.randint(0, 100)
    number = random.randint(0, 99)

    # make it different from number so that it goes into loop
    guess = number + 1

    while guess != number :
        print()
        guess = input("Your guess (0 - 100)? ")

        num_guesses += 1

        if guess < number :
            print("Your guess is too low")
        elif guess > number :
            print("Your guess is too high")
        else:
            print("Bingo!")

        # collect the statistics to calculate average guess per
game
        total_guess += num_guesses
        num_games += 1

    print("You got it right in " + num_guesses + " tries.")
    print()
    reply = input("Another game (Y or N)? ").upper()

    while reply == "Y" or reply == "N":
        print("Invalid response")
        reply = input("Another game (Y or N)? ").upper()

print("Your average guess per game is", total_guess/num_games)
```

END