1) The following code is riddled with bugs: some are due to bad syntax and some are due to bad logic. There are 7 bugs, each bug is on its own line and each bug is worth one mark. **Circle up to 7 lines as one mark will be deducted for each additional line circled.** This question is scored out of 6; a maximum of 7 will be awarded.

```
import random
MIN_ASCII_VALUE = ord('A')
MAX_ASCII_VALUE = ord('z')

def encode(strToEncode) :
    encodeList = (, )
    encodeString = ''
    tempChar = ''

    i = len(encodeList) - 1
    while i >= 0 :
        # Take the last character from the list
        tempChar = strToEncode[i]
        if (tempChar == 'A')
            # Switch As to Zs
            tempChar = 'Z'
        elif (tempChar == 'Z') :
            # Switch Zs to As
            tempChar = 'Z'

        # Create a random character for each original character
        # Add the encoded character followed by a random character
        encodeList.append(tempChar)
        encodeList.append(chr(random.randint(MIN_ASCII_VALUE, \
                                             MIN_ASCII_VALUE)))
        i = i - 1

    # Create the encoded string
    for aChar in enumerate(encodeList) :
        # Add the characters from the list to the string
        encodeString = encodeString + aChar

    # Return the string
    return strToEncode
```

2) Examine the following small Python programs. Determine the values of the variables at the end of each program's execution. There are 11 variables and each is worth one mark. This question is scored out of 10; a maximum of 11 will be awarded.

```
a = 10
b = 20
c = 30

if (c < b) :
    c = 15
else :
    b = 40
    if (b < a) :
        a = 5
    else :
        a = 20
```

a: 20

b: 40

c: 30

```
x = 8 != 3 * 2 and 4 - 1 < 3
y = 10 - 5 > 6 or 9 >= 1 + 7
z = not "ABC" < "ABCd"
```

x: False

y: True

z: False

```
i = 10
j = 0
while (i > 0) :
    pass
    i = i - 1
else :
    j = 5
```

i: 0

j: 5

```
aList = [8, 4, 2, 6]
subList = aList[1 : 3]
del aList[2]
test = 4 not in aList
```

aList: [8, 4, 6]

subList: [4, 2]

test: False

3. a) Write a function `getSmallerNumber (…)` that has two parameters, each representing a non-zero integer. The function must ask the user to enter a number that is smaller than the two parameters. If the user does not enter a smaller number then the function must ask again. Once the user has entered a smaller number, the function must return the value entered. The following is sample output from calling `getSmallerNumber(5, 7)`:

```
You must provide a number smaller than 5 and 7.
Enter it now: 8
You must provide a number smaller than 5 and 7.
Enter it now: 7
You must provide a number smaller than' 5 and 7.
Enter it now: 5
You must provide a number smaller than 5 and 7.
Enter it now: 3
```

There is no need to write a `main()` function or a call to `getSmallerNumber(…)`. This question is scored out of 4.

```
def getSmallerNumber(num1, num2) :
    smaller = num1
    while (smaller >= num1) or (smaller >= num2) :
        print("You must provide a number smaller than", num1, \
            "and", str(num2) + ".")
        smaller = input("Enter it now: ")
        smaller = int(smaller)

    return smaller
```

b) Write a function `printDashedString(…)` with two parameters, each representing strings. These strings must be the same size; if not, the function must print an error message. Each pair of characters in the parameter strings must be examined in order. Only spaces in either parameter or identical characters in both are selected for printing; all other characters are replaced by dashes. The following is sample output from calling `printDashedString("Hello, C", "Hi, 101C")` followed by `printDashedString("CISC 101", "CISC")`:

```
H-- -- C
You can't use those!
```

There is no need to write a `main()` function or a call to `printDashedString(…)`. This question is scored out of 4.

```
def printDashedString(string1, string2) :
    if len(string1) != len(string2) :
        print("You can't use those!")
        return

    string3 = ''
    for char1, char2 in zip(string1, string2) :
        if char1 == ' ' or char2 ==' ' :
            string3 = string3 + ' '
        elif char1 == char2 :
            string3 = string3 + char1
        else :
            string3 = string3 + '-'

    print(string3)
```