

Question 1. [8 MARKS]

Each subquestion on this page has a small piece of code that is supposed to work as described in the comment statement but has a small part missing. For each one, add the missing part inside the box. Your solution must follow the instructions in the comment statement. Each subquestion is independent.

Part (a) [2 MARKS]

```
d = {'not': 'me'}
```

```
# Make changes to the dictionary so that the code below the box prints "all better now".
```

```
# Do not reassign d to a new dictionary.
```

```
del d['not']

d['all'] = 'better now'
# a valid alternative is d['all better'] = 'now'
# or d['all better now'] = ''
# or d[''] = 'all better now'
```

```
for item in d.keys():
    print item, d[item]
```

Part (b) [2 MARKS]

Provide the while loop condition in the box so that the program works as described in the comments.

```
campus = ['UTM', 'StG', 'UTSC']
```

```
tries = 0
```

```
favourite = raw_input("Which campus do you like best? ")
```

```
# give the user at most 3 tries to pick one of the valid choices
```

```
while favourite not in campus and tries < 3:
```

```
    print "you must pick from UTM, StG or UTSC"
```

```
    favourite = raw_input("Pick again: ")
```

```
    tries += 1
```

```
if tries < 3:
```

```
    print 'Yes ' + favourite + ' is great!'
```

```
else:
```

```
    print "Can't you follow instructions?"
```

In the box beside each piece of code on this page, write its output. If it would generate an error, say so, and give the reason for the error.

Part (c) [1 MARK]

```
temp = 18
L = [9, temp, 27]
temp = 99
print L
```

[9, 18, 27]

Part (d) [1 MARK]

```
temp = [5, 10, 15]
L = [temp, [3, 6]]
temp[1] = 99
print L
```

[[5, 99, 15], [3, 6]]

Part (e) [2 MARKS]

```
L = [[1], [2], [3]]
L2 = L[:]
for element in L:
    element.append(8)
print L
print L2
```

[[1, 8], [2, 8], [3, 8]]
[[1, 8], [2, 8], [3, 8]]

Question 2. [6 MARKS]

In Assignment 2, we defined an **association list** to be a list of lists, such as

```
[ [3, ["hello", 27.4, True]], ["drama", [13, "comedy", "goodbye", 1]] ].
```

Each sublist has two elements: the first is called a “key” and is a value of any type, and the second is a list of values (of any type) that are associated with that key. No key occurs more than once in an association list.

Part (a) [1 MARK]

Consider the following function.

```
def keys_for(alist, v):  
    '''Return a list of all the keys associated with value v in association list alist.'''
```

Write a call to the function that should return a list of length 2.

Solution: There are many correct answers. Here is one.

```
keys_for([[1, [1, 2, 3]] , ['k2', [2, 5, 9]], ['k3', [5]]], 2)
```

Part (b) [5 MARKS]

Now write the function. You do not need to repeat the `def` line or the docstring.

Solution:

```
answer = []  
for item in alist:  
    key = item[0]  
    value_list = item[1]  
    if v in value_list:  
        answer.append(key)  
return answer
```

Question 3. [5 MARKS]**Part (a)** [4 MARKS]

As you know, a string can contain newline characters. When it does, we think of the string as having multiple lines within it.

Write the following function according to its docstring.

```
def line_lengths(s):  
    '''Return a list of ints containing the length of each line in string s. The newline  
    character at the end of a line does not count toward the line's length.'''
```

Solution:

```
lines = s.split("\n")  
lengths = []  
for line in lines:  
    lengths.append(len(line))  
return lengths
```

Part (b) [1 MARK]

Write a main block that will use your function to determine the lengths of the lines in file `poem.txt` and then print the list that your function produces.

Solution:

```
if __name__ == '__main__':  
    s = open('poem.txt').read()  
    result = line_lengths(s)  
    for line in result:  
        print line
```