Name:

_____

_____

Student Number:

_____

University of Cape Town ~ Department of Computer Science

Computer Science 1015F ~ 2013

Class Test 3

** *SOLUTIONS* **

Question	Max	Internal	External
1	12		
2	9		
3	9		
TOTAL	**30**		

Marks : 30

Time : 45 minutes

Instructions:

a) Answer all questions.

b) Write your answers in **PEN** in the spaces provided.

c) Show all calculations where applicable.

Question 1 [12]

Examine the following function:

```
def puzzle(firstList, secondList, value):
      dict = {}
      for i in range(len(firstList)):
          dict[firstList[i]] = secondList[i]
      for key in dict:
           if dict[key] == value:
               return key
```

(a) Explain in simple English what this function does. [2]

Returns the element in the first list at the position that corresponds to where value is found in the second list. [2]

(b) What is the value returned by:

`puzzle(['a', 'b', 'c'], [100, 200, 300], 200)` ? [1]

'b' [1]

(c) What is the value returned by:

`puzzle(['a', 'b', 'c'], [100, 200, 300], 400)` ? [1]

None [1]

(a) What kinds of parameters as input would cause this function to fail, with a Python error message. Explain why the function will fail and provide example parameters. [3]

If firstList is longer than secondList [1] then this will cause an out of range error in the first loop as the index will exceed the length of the second list. [1] Example parameters: firstList = ['a', 'b', 'c'], secondList = [100,200], value = 200 [1].

(b) Rewrite the code for the function `puzzle` so that it doesn't need to search through the entire dictionary. [5]

```
def puzzle(firstList, secondList, value):
    dict = {}                                [1]
    for i in range(len(firstList)):          [1]
        dict[secondList[i]] = firstList[i]   [1]
    if  value in dict:                       [1]
        return dict[value]                   [1]
```

Question 2 [9]

Examine the following function:

```
def rec(n):
    if n==0:
        return 0
    else:
        return rec(n-1) - n
```

(a) Identify the base case and recursive step in this function. [2]

base case – n = 0 [1]

recursive step (n > 0) call to rec(n-1) [1]

(b) What is the value returned by `rec(3)` ? [2]

-6 [2]

(c) What would happen if the line "return rec(n-1) - n" was replaced by "return rec(n) – n" ? [1]

Python would recurse until the recursion depth was exceeded [1] (Also accept that this represents infinite recursion [1])

(d) Rewrite this function to use iteration rather than recursion [4]

```
def rec(n):
  negsum = 0          [1]
  for i in range(n):  [1]
    negsum -= n       [1]
  return negsum       [1]
```

Question 3 [9]

Examine the following function:

```
def doSomething(lst,value):
    p = 0
    while p < len(lst) and lst[p] != value:
        p += 1
    if p == len(lst):
        return -1
    else:
        return p
```

(a) What is this function more commonly called [1]

Linear Search [1]

(b) What is the time complexity of this function in the worst case? [1]

Linear – O(n) [1]

(d) What is the value returned by:

`doSomething([7, 14, 21, 14, 35], 9)` ? [1]

-1 [1]

(c) Modify the function `doSomething` so that it finds the position of the last occurrence of value in the list. [6]

_____
_____
_____
_____
_____
_____
_____
_____
_____

```
    def doSomething(lst, value):
        p = len(lst)-1    [1]
        while p >= 0 and lst[p] != value: [2]
            p -= 1       [1]
        return p         [2]

(or instead of last line
if p == -1:
    return -1
else
    return p)
```