University of Cape Town ~~ Department of Computer Science

Computer Science 1015F ~~ 2016

Class Test 3 - Makeup

** *Solutions* **

Enter the following details AND shade in the corresponding blocks to the right with your Student Number.

Faculty (please tick one):

Science	Engineering	Commerce	Humanities	Other:

Student Number :

_____

Name (optional) :

_____

Marks : 40

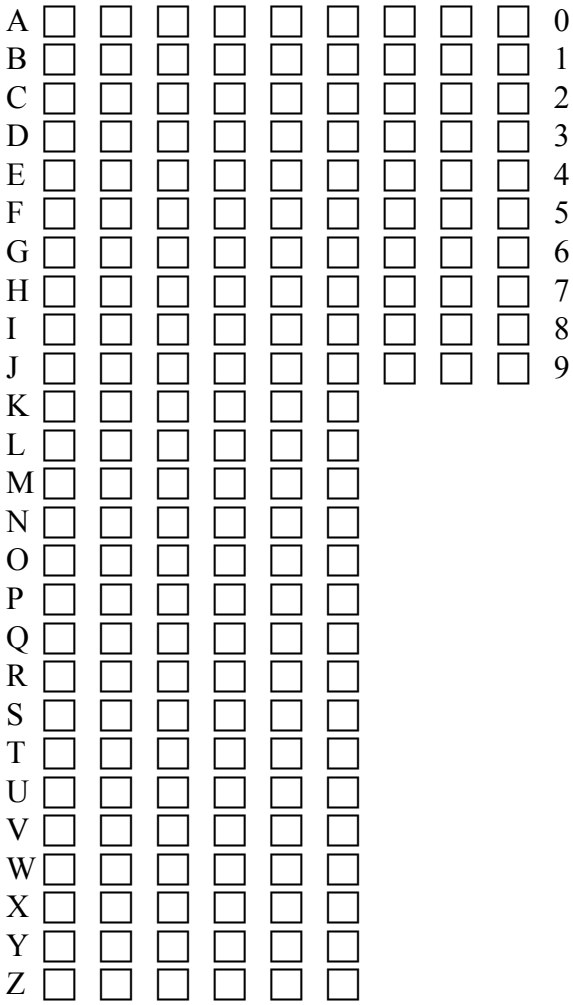
Time : 40 minutes

Instructions:

a) Answer all questions.

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

c) Show all calculations where applicable.



FOR OFFICIAL USE ONLY:

Question	1	2	3	4	5	6	7	8
Max	23	17						
Marks								
Marker								

Question 1 – Arrays, Dictionaries and Files [23]

Examine the `Q1.py` module listed at the end of the test and answer the following questions.

(i) Explain what happens in the `xWords(wordList)` function if:

A. the file "`input.txt`" does not exist in the current directory. [1]

Ans error occurs when opening the file and the program will crash [1] (IOError)

B. the file "output`.txt`" does not exist in the current directory. [1]

It will be created in the currently directory[1]

(ii) Describe briefly, and in clear English, what the function `xWords(wordList)` does and what output is produced. Your answer must consider parameters of different types. [5]

This function opens a file called "input.txt? And writes to a file called "output.txt".[1] If the wordList parameter is a dictionary, all the words in input.txt are checked to see if they are in the dictionary. If they are, the word is replaced with the dictionary value. [2] Tall the words are then written to the output.txt file [1] Otherwise, no values are written to the file – pattern.txt will be empty.[1]

(iii) Write down the exact output when `Q1.py` is run in the in the Python3 interpreter. [6]

[[0], [0, 0], [0, 0, 0]] #[2]
[['x', 'x'], [1, 1, 1], ['a', 'a', 'a', 'a']] #[2]
[] #[2]

(iv) Write the missing code for the `getAllLoc2Arr(arr2D,value)` function in the `Q1.py` module. This function returns **all** the locations as a list where an item, `value`, appears in the a 2D array, `arr2D`. For example, the following code:

```
def getAllLoc2Arr(arr2D,value):  #add in missing code
    pos=[] #[1]
    for i in range(len(arr2D)): #[2]
        for j in range(len(arr2D[i])): #[2]
            if arr2D[i][j]==value:#[2]
                pos.append([i,j]) #[2]
    return pos #[1]
```

Question 2 - Recursion [17]

Examine the `test3_Q2_2016.py` module listed on the last sheet of the test and answer the following questions.

(i) Write down the **exact output** when this module is executed (e.g. when the user presses the "Run" button in Wing101)? [2]

```
['e', 'g']     # [1] mark
[]             # [1] mark
```

(ii) Write a recursive version of `test3_Q2_2016.py` [7]

```
One correct answer is:
def someRec(s):
        if len(s) <= 1:  #[1]
            return []  #[1]
        else: #[1]
            l = someRec(s[1:-1]) #[1]
            if s[0] == s[-1]: #[1]
            l.insert(0, s[0]) #[1]
        return l #[1]
```

(iii) The recursive approach to calculating Fibonacci numbers (as listed below) is much slower than the iterative approach. Explain why and provide an example to support your explanation. [3]

```
def fibRec(x, n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibRec(n-1)+fibRec(n-2)
```

There are many duplicated function calls which end up recalculating the same Fibonacci number repeatedly [1]. For example: fib(4) calls fib(3) and fib(2). fib(3) calls fib(2) and fib(1). In this case fib(2) is called twice. The problem is worse for higher values of n. [2] (Other example acceptable as long as it shows the duplicate function calls.)

(iv) Write an iterative version of the recursive Python program below: [5]

```
def strangeRec(s):
    if s == '':
        return []
    else:
        l = strangeRec(s[:-1])
        l.append(ord(s[-1])
        return l
print(strangeRec('Hello'))
```

One solution is:

```
def strangeIt(s): #[1]
   olist = []   #[1]
   for c in s:  #[1]
      olist.append(ord(c)) #[1]
  return olist #[1]
```

Code examples for the test – you may detach this sheet.

```
    #Q1.py

    def xWords(wordList):
        if type(wordList)==type({}):
            f1=open("input.txt",'r')
            f2=open("output.txt",'w')
            for line in f1:
                words=line.split()
                for ind in range(len(words)):
                    if words[ind] in wordList:
                        words[ind]=wordList[words[ind]]
                print("  ".join(words),  file=f2)  #  join
    converts list->string
            f1.close()
            f2.close()


    def arrFrmt(values,init):
        result=[]
        for val in values:
            if type(val)==type([]):
                row=[]
                for i in range(val[1]):
                    row.append(val[0])
                result.append(row)
            else:
                result.append([0]*val)
        return result


    arr1=[1,2,3]
    arr2=[['x',2],[1,3],['a',4]]
    arr3=[]

    x=arrFrmt(arr1,0)
    print(x)
    x=arrFrmt(arr2,0)
    print(x)
    x=arrFrmt(arr3,0)
    print(x)

def getAllLoc2Arr(arr2D,value):  #add in missing code
```

Code examples for the test – you may detach this sheet.

```
# test3_Q2_2016.py
def someIt(s):
     olist = []
     for i in range(len(s) // 2):
         if s[i] == s[len(s)-1-i]:
             olist.append(s[i])
     return olist


print(someIt('begger'))
print(someIt('X'))
```