University of Cape Town ~~ Department of Computer Science

Computer Science 1015F ~~ 2018

Class Test 3

** *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 : 35

Time : 40 minutes

Instructions:

a) Answer all questions.

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

c) Show all calculations where applicable.

A ☐☐☐☐☐☐☐☐☐ 0
B ☐☐☐☐☐☐☐☐☐ 1
C ☐☐☐☐☐☐☐☐☐ 2
D ☐☐☐☐☐☐☐☐☐ 3
E ☐☐☐☐☐☐☐☐☐ 4
F ☐☐☐☐☐☐☐☐☐ 5
G ☐☐☐☐☐☐☐☐☐ 6
H ☐☐☐☐☐☐☐☐☐ 7
I ☐☐☐☐☐☐☐☐☐ 8
J ☐☐☐☐☐☐☐☐☐ 9
K ☐☐☐☐☐☐
L ☐☐☐☐☐☐
M ☐☐☐☐☐☐
N ☐☐☐☐☐☐
O ☐☐☐☐☐☐
P ☐☐☐☐☐☐
Q ☐☐☐☐☐☐
R ☐☐☐☐☐☐
S ☐☐☐☐☐☐
T ☐☐☐☐☐☐
U ☐☐☐☐☐☐
V ☐☐☐☐☐☐
W ☐☐☐☐☐☐
X ☐☐☐☐☐☐
Y ☐☐☐☐☐☐
Z ☐☐☐☐☐☐

FOR OFFICIAL USE ONLY:

Question	1	2	3	4	5	6	7	8
Max	15	10	10					
Marks								
Marker								

Marks (for each question)

0 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
1 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
2 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
3 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
4 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
5 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
6 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
7 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
8 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐
9 ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐ ☐☐

Question 1 – Testing and Arrays [15]

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

(i) Does the function call

```
do_a([1,2,3],0)
```

comprise a complete **statement coverage test** of the `do_a` function? Explain your answer.[2]

No. #[1 mark]
The lines
result=True
break
have not executed. #[1 mark]

(ii) Write down a set of function calls that will comprise a **complete**, but **minimal**, path test of the function `do_a`. [2]

you need three function calls:
one with empty list, e.g. `do_a([],0)`
one with item in list, e.g. `do_a([1,2,3],1)`
one with item not in list, e.g. `do_a([1,2,3],0)`
[2 marks if all correct, 1 mark if two correct function calls, otherwise 0]

(iii) Write down the **exact output** produced when the `Q1.py`. module is executed. [5]

#each line has to be absolutely correct in order to get a mark for the line – no half marks
t #[1 mark – checks understanding of array access]
4 #[1 mark – checks understanding of function]
c #[1 mark – checks understanding of character sorting]
[1, 3, 4, 15, 30] #[1 mark – checks understanding that arrays passed by reference]
[[0, 1, 2], [1, 2, 3], [2, 3, 4]] #1 mark – checks understanding of do_c function gradient fill

(iv) Write Python code for the function `do_d(lst1,lst2)` in the `Q1.py` module. This function **returns a list** that is the result of subtracting the elements in `lst1` from the corresponding elements in `lst2`. The lists must be of the same length; if they are not, `do_d` returns an empty list. For example, in the Python interpreter, `do_d` would behave as follows.

```
>>> do_d([4,5,6],[0,4,0])
    [4, 1, 6]
>>> do_d([4,5,6],[1,8,3])
    [3, -3, 3]
>>> do_d([4,5,6],[0,4])
    []
```

Now complete the function below. [6]

```
def do_d(lst1,lst2)
```

```
def do_d(lst1,lst2):
    ans=[] #[1]
    if len(lst1)==len(lst2):#[1]
        for i in range(len(lst1)): #[1]
            tmp=lst1[i]-lst2[i] #[1]
            ans.append(tmp) #[1] - can obv. be combined with line above
    return ans#[1]
```

Question 2 – Recursion [10]

Examine the `Q2.py` module listed at the end of the test and answer the following questions.

(v) The function `do_rec1` in the `Q2.py` module function is recursive. What is the recursive base case for this function? [1]

if s=='':return 0
OR "when the string is empty" or equivalent.

(vi) Explain clearly, briefly and in general terms, what the `do_rec1` function does. [1]

It counts the number of 'c' characters in the string 's'. (must be in general terms)

(vii) Write down the exact output when the `Q2.py` module is run in the Python interpreter. [4]

4 #[1 mark]
0 #[1 mark]

1
f2
fl3
flo4 #[2 marks]

(viii) Now write a **recursive** function `tot(minimum,maximum)` that will return the sum of all the numbers from the minimum to the maximum specified (the sum must include the minimum and maximum). For example, in the Python interpreter, `tot` would behave as follows.

```
>>>tot(2,4)
    9
>>>tot(4,4)
    4
>>>tot(5,3)
    0
```

Now complete the function below. [4]

```
def tot(minimum,maximum):
    """Returns the sum of the numbers from m to n"""
```

_____
_____
_____
_____
_____
_____
_____
_____
_____

def tot(minimum,maximum):
"""Returns the sum of the numbers from m to n"""
if minimum <=maximum: #[1]
*return minimum+**tot(minimum+1,maximum)** #[2]*
return 0 #[1]

*Note: **zero marks if recursive call missing!!** Also., function can be written to count down instead of up.*

Question 3 – Dictionaries and Files [10]

Examine the `Q3.py` module listed at the end of the test and answer the following questions.

(i) Explain what happens if this module is executed and a file called "`inp.txt`" does not exist in the current folder. [1]

The program crashes (with a IOError)

(ii) This module contains an example of a *tracing statement*. Which statement is the *tracing statement?* [1]

print(word,weight)

(iii) The file "`inp.txt`" contains the following lines of text.

```
Jack

and jill

And bob

ran up the hill
```

Write down the exact contents of the file "`out.txt`" after the program has executed. [3]

```
2 1

3 5

4 3
```

#1 mark for having 3 lines + #2 marks if all correct, 1 mark if only one mistake

(iv) Now write a function `create_d(key_lst,val_lst)` that returns a dictionary with the specified `key_lst` elements as the dictionary keys and the values in `val_lst` as the corresponding values for the keys. For example, in the Python interpreter, this function behave as follows.

```
>>>create_d(['a','b','c'],[4,5,6])
   {'a': 4, 'b': 5, 'c': 6}
>>>create_d(["ogre","troll","goblin"],[400,50,60])
   {'goblin': 60, 'ogre': 400, 'troll': 50}
>>>create_d([2,56],["happy",80])
   {2: 'happy', 56: 80}
```

Now complete the function on the following page. [5]

```
def create_d(key_lst,val_lst):
```

def create_d(key_lst,val_lst):

```
    out={} #[1]

    for i in range(len(key_lst)): #[1]

        out[key_lst[i]]=val_lst[i] #[2]

    return out #[1]

#or alternative correct answer
```

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

Question 1

```
#Module Q1.py
def do_a(arr1,item):
    result=False
    if arr1!='':
        for a in arr1:
            if a==item:
                result=True
                break
    return result

def do_b(lst):
    lst.sort()
    med=len(lst)//2
    return lst[med]

def do_c(a,b):
    z=[]
    for i in range(a):
        z.append([])
        for j in range(b):
            z[i].append(i+j)
    return z


def do_d(lst1,lst2):
    """This function returns a list that is the
result of subtracting lst1 from lst2"""
    #Code missing here

chars=['c','a','t']
X=[30,3,15,4,1]
print(chars[2])
val=do_b(X)
print(val)
val=do_b(chars)
print(val)
print(X)
print(do_c(3,3))
```

Question 2

```
#Module Q2.py
def do_rec1(s,c):
    if s=='':return 0
    if s[0]==c:
        return 1 + do_rec1(s[1:],c)
    return do_rec1(s[1:],c)

def do_alt_rec2(s):
    n=len(s)
    if n>0:
        do_alt_rec2(s[:-1])
        output=s[:-1]+str(n)
        print(output)

print(do_rec1("mississippi",'i'))
print(do_rec1("huckleberry",'a'))
print()
do_alt_rec2("flow")
```

Question 3

```
#Module Q3.py

file1=open("inp.txt",'r')
lines=file1.readlines()
file1.close()
weights={}

for line in lines:
    line=line.strip('\n')
    words=line.split()
    for word in words:
        weight=len(word)
        print(word,weight)
        if weight not in weights:
            weights[weight]=1
        else:
            weights[weight]+=1

file2=open("out.txt",'w')
for w in sorted(weights):
    print(w,weights[w],file=file2)
file2.close()
```