University of Cape Town ~~ Department of Computer Science

Computer Science 1015F ~~ 2014

Supplementary Class Test 2

** *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

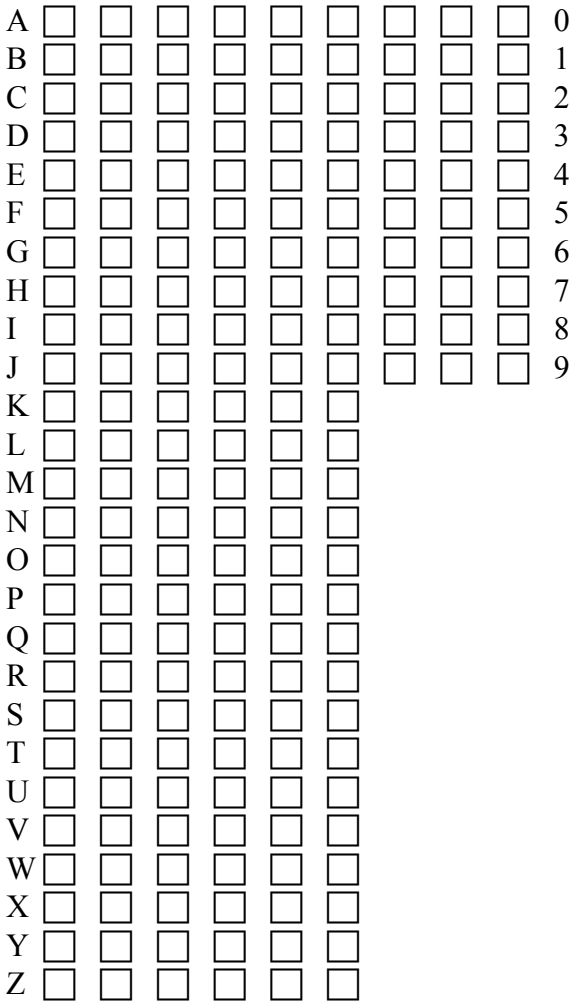
Time : 45 minutes

Instructions:

a) Answer all questions.

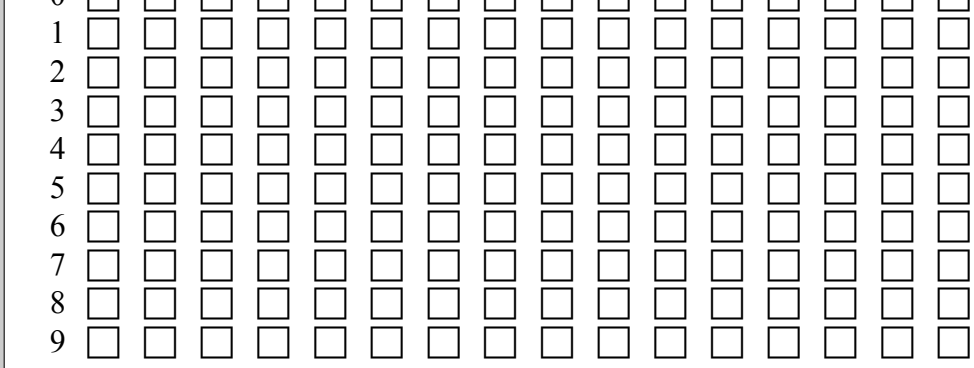
b) Write your answers in PEN in the spaces provided.

c) Show all calculations where applicable.



FOR OFFICIAL USE ONLY:

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



Question1 [15]

Examine the `Q1.py` module listed on the last sheet of the test and answer the following questions.

(i) What does the Python interpreter do to the code in `Q1.py` when the user runs the module? [2]

The interpreter translates the code one line at a time [1] into machine code [1], which is executed on the computer. [1] #any two points

(ii) From the module `Q1.py,` give an example of a **builtin** Python function that returns a value and a function that does not return a value. [2]

A. Builtin function in module `Q1.py` that **returns** a value:

max() #[1]

B. Builtin function in module `Q1.py` that **does not** return a value:

print() #[1]

(iii) Write down the exact output of the `Q1.py` module if the user runs the module. [6]

Happy #[1]
FELLA #[3]
Happy #[1] - this tests understanding of scope
!!! #[1]

(iv) Rewrite the code for the boolean function `subset(wrdA,wrdB)` so that it works as follows. This function should **return True** if **all** the charaters of `wrdA` can be found in `wrdB`. For example (in the Python3 interpreter):

```
>>>subset("bets","beetles")
>>>True
>>>subset("bats","beetles")
>>>False
>>>subset("steel","beetles")
>>>True
```

[5]

```
#there are many ways to do this - clever tricks OK.
def subset(wrdA, wrdB):
    for i in wrdA: #[2]
        if i not in wrdB:#[1]
            return False#[1]
    return True #[1]
```

Question2 [10]

Study the following program to count the number of occurrences of an item in a list:

```
def count (values, item):
    counter = 0
    for i in range (len (values)):
        if values[i] == item:
            counter += 1
    return counter

item = int (input ("Enter an item:\n"))
print (count ([1,2,3,4], item))
```

(i) Suppose that we are using equivalence classes to test the program. Describe equivalence classes and boundary values that can be used when testing the function. DO NOT provide test values – only descriptions. [4]

either ... exists within list [1]; does not exist [1]; as first item [1]; as last item [1]
or ... does exist [1]; does not exist [1]; exactly 0 occurences [1]; exactly 1 occurrence [1]

(ii) Provide a set of test values that will test this program when using statement coverage. [1]

any value within list [1]

(iii) How many test values are needed when using exhaustive testing? [1]

infinitely many [1]... OR as many as there are integer values [1]

(iv) If line 4 referred to **values[i+1]** instead of **values[i]**, what type of error is this: logic or syntax? Does it get detected at compile-time or at run-time? [2]

logic [1]; run-time [1]

(v) What are 2 techniques that may be use by a programmer to find the cause of logic errors in a program? [2]

trace statements [1]; using a debugger [1]

Question3 [10]

Count functions that are able to count occurrences of items in a list are built into many programming languages. With reference to the count function provided in the previous question, answer the following questions.

(i) Write a count function that counts all items in the given list BUT the supplied item. Assume that the list and the item are passed in as parameters. (Hint: modify the program in Question 2) [3]

one possible solution:

```
def count2 (values, item):
   counter = 0
   for i in range (len (values)):
      if values[i] != item:
         counter += 1
   return counter
```

(ii) Write a generalized count function that behaves as follows. Suppose you are given 2 lists – listA and listB. Your function must indicate how many times each item in listB appears in listA. Your function should take 2 lists as parameters and print <item, count> pairs to the screen. For example: [7]

```
count_all ([1,2,4,3,4,2,6], [1,2,3,4,5,6])
1 1
2 2
3 1
4 2
5 0
6 1
```

_____

_____

_____

one possible solution (assume they may use any algorithm ... and use negative marking: -1 for each error in their program)

```
def count_all (values, items):
    for item in items:
        counter = 0
        for i in range (len (values)):
            if values[i] == item:
                counter += 1
        print (item, counter)
```

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

Question 1

```
#Q1.py
def jam(wordA,wordB):
     if not wordA.isalpha() and wordB.isalpha():
          return "!!!"
     wordA=wordA.lower()
     wordB=wordB.upper()
     while wordB:
          big=max(wordA)
          rep=max(wordB)
          wordA=wordA.replace(big,rep)
          wordB=wordB.replace(rep,'')#''=empty string
     return wordA

def subset(wrdA, wrdB):
     pass  #function to be rewritten

wordA="Happy"
wordB="fleas"
print(wordA)
y= jam(wordA,wordB)
print(y)
print(wordA)
print(jam("Have2Have",wordB))
```