

University of Cape Town ~~ Department of Computer Science

Computer Science 1015F ~~ 2015

Class Test 2 draft

***** Solutions *****

Consider the following main program:

Write the test_module **module**, containing the `middle` function. This function calculates the middle value (i.e., not the maximum or minimum) of 3 numbers. Hint: Use the built-in `min` and `max` functions.

This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

2

Question 2 - Hardware and Software [9]

1. What is the difference between a high level language and a low level language? Why do programmers prefer to use high level languages? [3]

high - easier for humans to understand [1]; low - easier for machines to understand [1]; programmers are more productive and make fewer errors when using a language geared to their understanding [1]

2. Give 2 examples of high level languages and 2 examples of low level languages. [2]

python, java, c++, etc. [1]; machine language + assembler [1]

3. Python is an interpreted language. Discuss 1 key advantage and 1 disadvantage of interpreted languages like Python. [2]

adv: faster to code - no compile step [1]; disad: slow to execute [1]

4. In Von Neumann's architecture of a computer, what elements did he include besides the Control Unit and Arithmetic Logic Unit? [2]

memory [1] input/output [1]

Write your own version of the Python `replace` function that executes a search-and-replace function on a string and returns this transformed string.

Your version should take 3 arguments: the source string, the string to search for and the replacement string.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

*[-1 for leaving out return;
6 = completely correct code
5 = only one minor error
4 = 2 minor errors
3 = has some idea but code will not work
2 = only vague idea in code
1 = mostly unusable but maybe one sensible statement somewhere in solution
0 = nothing that resembles even a correct statement]*

any algorithm that provides a correct answer is correct. watch out for algorithms that do in-place replacements so a search-and-replace on "X"/"XX" will result in an infinite loop.

Question 4 – Testing [10]

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

The function `Rev(lst)` in `MK.py` reverses the items in a list.

- a) You have been asked to test the function `Rev(lst)` using either an **exhaustive testing** or a **random testing** strategy. Which method do you choose and why? [2]

random testing (1 mark) . Exhaustive testing would require infinitely many input values (1 mark) – all possible lists.

- b) Briefly describe the equivalence classes and boundary values that can be used when testing the function `Rev(lst)`, **giving an example of each.** [3]

Equivalence classes:

full list e.g. [1,2,4,5,6]; (1 mark)

empty list e.g. [] (1 mark)

(can also have non-list – not required, but have mark if included with example e.g. 'hi')

Boundary values:

list with one element e.g. [2] (1 mark)

- c) Write down a set of test values for **statement coverage** testing of function `Rev(lst)`. [1]

any non-empty list (1 mark)

- d) Write down a set of test values for **path coverage** testing of function `Rev(lst)`. [2]

an empty list (1 mark)

any non-empty list (1 mark)

-1 mark if too many items given (but no negative marking)

e) The Python program below will generate an error:

```
import MK.py
```

```
MK.Rev(2)
```

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]*

Question 5 – Arrays [15]

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

- a) Describe briefly, and in clear English, what the function `Mystery(lst)` outputs. [2]

The middle element of a list.

If the list has an even number of elements, it outputs the lower one [1]

- a) Write down the exact output of the `MK.py` module if the user runs the module. [8]

[10, 8, 6, 4, 2] #2 marks – the easiest to get in this question

Private #2

6 #2

$[[5, 6], [3, 4], [1, 2]]$ #2

- b) Rewrite the code for the function `Mystery(lst)` so that it works as follows. This function should **return True** if every element in the list is unique (i.e. there are no duplicates). For example (in the Python3 interpreter):

```
>>>Mystery(["bets", "beetles", "bats"])
```

```
>>>True
```

```
>>>Mystery(["bats","beetles","bats"])
```

```
>>>False
```

Complete the code below: [5]

```
def Mystery(lst):
```

[illegible]

One correct answer is:

```
def Mystery(lst):  
    for i in lst: # 1 mark  
        if lst.count(i)>1: #2 mark  
            return False #1 mark  
    return True #1 mark
```

Another correct answer is:

```
def Mystery(lst):  
    for i in lst: #1  
        count=0  
        for j in lst: #1  
            if j==i:  
                count+=1 #1  
        if count>1:  
            return False #1  
    return True #1
```


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

Question 4 and 5

```
# MK.py

def main():
    list1=[2,4,6,8,10]
    list2=["Skipper","Kowalski","Rico","Private","King Julian"]
    list3=[[1,2],[3,4],[5,6]]

    print(Rev(list1))
    print(Mystery(list2))
    print(Mystery(Rev(list1)))
    print(Rev(list3))

def Rev(lst): #reverses the items in lst
    tmp=[]
    for i in range(len(lst)-1,-1,-1):
        tmp.append(lst[i])
    return tmp

def Mystery(lst):
    res=''
    if lst!=[]:
        lst.sort()
        md=len(lst)//2
        res=lst[md]
    return res

main()
```