

Please fill in your Student Number and Name.

Student Number : _____

Name:

Student Number:

University of Cape Town ~ Department of Computer Science

Computer Science 1015F ~ 2017

June Examination

**** SOLUTIONS ****

Question	Max	Internal	External
1	20		
2	10		
3	10		
4	10		
5	10		
6	10		
TOTAL	70		

Marks : 70

Time : 120 minutes + 10 minutes reading time at the start

Instructions:

- Answer all questions.
- Write your answers in PEN in the spaces provided.
- You may use a calculator, BUT show all calculations where required.

Question 1 [20]

For each question below, **write down ONE letter** corresponding to the **most correct** answer.

Consider this Python function for Questions a and b below:

```
def score(word):  
    x = len(word)  
    for r in range(0,x):  
        print (word[r])  
    return x
```

a. What is an example of a string literal?

- a) x
- b) word
- c) word[r]
- d) None of the above.

d

b. What is an example of a reserved word?

- a) range
- b) for
- c) print
- d) word

b

c. What string is generated by 'science'[0::2]?

- a) see
- b) ee
- c) sine
- d) in

c

d. What numbers are generated by range (7,1,-3)?

- a) 7,4
- b) 7,4,1

c) -3,-2,-1,0,1,2,3,4,5,6

d) -3

a

e. Which command below will exit a loop immediately?

- a) continue
- b) break
- c) pass
- d) jump

b

f. Which of the following will generate a string with the word "hi!" centred in fixed width?

- a) "{0:^20}".format("hi!")
- b) "{0:>20}".format("hi!")
- c) "{0:<20}".format("hi!")
- d) "{0:*20}".format("hi!")

a

g. Which of the following is writable storage that maintains values without power?

- a) RAM
- b) VRAM
- c) CPU cache
- d) Hard drive

d

h. Which type of code is easier for a human to understand?

- a) Machine code

- b) Low-level language
- c) High-level language
- d) Assembly

c

i. Consider this Python function:

```
def Maximum(x, y):
    z=x
    if (x<y) z=y
    return z
```

The function contains an error. This error is:

- a) A **syntax** error.
- b) A **run-time** error.
- c) A **logic** error.
- d) All of the above.

a

j. The following is an example of a **Glass Box** testing method:

- a) Equivalence Classes
- b) Path Coverage Testing
- c) Random Testing
- d) Exhaustive Testing
- e) None of the above

b.

k. Given the Python function:

```
def Minimum(x, y):
    z=x
    if (x>y): z=y
    return z
```

The function call

Minimum(3, 2)

constitutes

- a) A complete **path coverage test**.
- b) A complete **statement coverage test**.
- c) All of the above.
- d) None of the above

b.

l. What output does the Python code below produce?

```
X=[['a', 'b', 'c'], [3, 4, 5],
    [6, 7, 8]]
```

```
print(X[1][2])
```

- a) IndexError
- b) ['a', 'b', 'c']
- c) b
- d) 5
- e) 8

d.

m. What output does the Python code below produce?

```
X=['b', 'a', 't']
```

```
X.sort()
```

```
print(X[2])
```

- a) IndexError
- b) ['a', 'b', 't']
- c) 'b'
- d) 't'
- e) []

d

n. The Python code below outputs

```
f=open("data.txt", 'r')
x=f.readlines()
```

```
print(len(x))
```

```
f.close()
```

a) Nothing.

b) The number of characters in the first word in the file "data.txt".

c) The number of characters in the first line of the file "data.txt".

d) The number of lines in the file "data.txt".

e) The number of characters in the file "data.txt".

a) $O(1)$

b) $O(\log n)$

c) $O(n \log n)$

d) $O(n)$

e) $O(n^2)$

e

d.

o. Which of the function definitions below will produce this output?

```
myst("hello")
```

```
o
```

```
lo
```

```
llo
```

```
ello
```

```
hello
```

a) `def myst(s):`

```
    if s!='': myst(s[1:])
```

```
    print(s)
```

b) `def myst(s):`

```
    print(s)
```

```
    if s!='': myst(s[1:])
```

c) `def myst(s):`

```
    for a in s: print(a)
```

d) `def myst(s):`

```
    for i in range(len(s)):
```

```
        print(i)
```

e) None of the above.

a.

q. What does it mean if a sorting algorithm is *stable*?

a) It uses very little additional memory.

b) It is very fast relative to other sorting algorithms.

c) It maintains the relative order of items with equal values.

d) It does less work if the list is already sorted.

e) It sorts the list in reverse order.

c.

r. 10.11_2 is equal to:

a) 4.22_{10}

b) 4.3_{10}

c) 2.75_{10}

d) 2.3_8

e) $1A_{16}$

c

s. Given the following truth table, what is the Boolean expression F represented by?

A	B	C	F
0	0	0	0

p. What is the time complexity of the **Quicksort** algorithm in the **worst case**?

0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

0 01111111 000000000000000000000000

a) infinity

b) 0

c) 1

d) 0.5

e) -1.25×10^{128}

a) **A OR B**

b) **B AND C**

c) **A NAND B**

d) **(A AND B) OR (B AND C)**

e) **A AND B AND C**

c

d

- t. What is the decimal floating point value of the following IEEE 754 single precision number?

Question 2 [10]

Examine this program and answer the first three questions in this section.

```
def two(a,b,c):  
    if a>b and b>c:  
        print (2*a)  
    elif a<b:  
        print (2*c)  
    else:  
        print (2*b)  
  
two (2, 3, 'x')  
  
two (3, 1, 2)
```

- a. Write down the **exact output** to the screen when this module is executed. [2]

xx

2

#1 mark for each line

- b. Assuming the invocation of the two methods remain unchanged, what would happen if you replaced the boolean operator **and** in the function `two` with an **or** operator? *Explain why.* [2]

there would be a runtime error[1] during the first invocation of the two method because the first term in the boolean expression would be true, causing it to evaluate the second term where we would be comparing a str to an int [1]

- c. Write code to replace/encapsulate the final two lines of code so that this program can be used as a module or standalone program. [2]

```
def main():
    two(2,3,'x')
    two(3,1,2)

if __name__ == "__main__":
    main()
```

d. Rewrite the following for loop as a while loop.

```
for i in range(a, b, 2):
    sum += i
```

[2]

```
i=a                # [1/2] for starting
while i < b:        # [1/2] for loop constraint
    sum = sum + i   # [1/2] for sum update (+= is fine too obviously)
    i += 2          # [1/2] for loop var update
```

e. Explain the difference between the following two lines:

```
x = a//5
x = a/5
```

[2]

*returns int using floor division [1] (½ if the omit the fact that it drops any decimal instead of rounding. They can say “integer division” “floor division” or the like
returns a float using normal division.*

Question 3 [10]

- a. Python provides a string function called `str.upper()` that will convert all alpha characters to uppercase. Your task is to implement a custom version, called `my_str_upper()`, that will take a string as an argument and return that string in uppercase (capitals), using the `chr()` and `ord()` methods. Make sure to document your function correctly. You CANNOT use any built-in string functions (e.g., `str.upper()`), but rather use first principles and manipulate strings at the character level. [6]

```
def my_str_upper (sentence):
```

```
# We did the lowercase version of this in class
def my_str_upper(sentence):
    """ Re-implement the string.upper() method """ # [1] for docstring
    out = ""
    lwr_to_upr = ord('A') - ord('a') # [1] for correctly calculating offset
    for c in sentence:
        if ord(c) >= ord('a') and ord(c) <= ord('z'): # [1] for the bounds checking
            out += chr(ord(c) + lwr_to_upr) # [1] for converting case
        else:
            out += c # [1] for keeping old characters in tact
    return out # [1] for returning and not printing
```

- b. Assume that the function from the previous question was implemented correctly. Write a program that asks the user for a sentence, uses the `my_str_upper()` method to convert the sentence to uppercase and prints it, and then asks for another sentence. The program should continue until the user enters the sentence “q”. [4]

```
# wrapping it in main() and making it a module is optional.
sentence = input("Enter a sentence ('q' to quit): ")          # [1] for initial input call
while sentence != 'q':                                       # [1] for correct loop and stop
    print (my_str_upper (sentence))                          # [1] for print and method use
    sentence = input("Enter a sentence ('q' to quit): ")    # [1] for 2nd input inside loop
```

Question 4 [10]

Examine the **Q4.py** module and the contents of the files `happy.txt`, `exam.txt`, and `values.txt`, which are listed at the end of the exam paper. Then answer the following questions.

- a. Write down the **exact output** to the screen when this module is executed, assuming that the `bFind` function operates correctly. [2]

```
[[80, '30', '20'], ['c', 'd', 'f', 'a', 'b'], ['10', '3', '2', '7', '1']]
```

```
[[20, '30', '80'], ['a', 'b', 'c', 'd', 'f'], ['1', '10', '2', '3', '7']]
```

#1 mark for each line

#Note: do not penalize for not having the numbers in quotes, i.e. [20,30,80] is okay.

- b. Write down the **exact contents** of the `exam.txt` file after this module is executed, assuming that the `bFind` function operates correctly. [3]

```
20 0
```

```
c 2
```

```
50 -1
```

#1 mark for each correct line

#-1 if previous contents are listed.

- c. The function `bFind` is missing some lines. Write down the hidden lines here, so that the function will operate correctly. [3]

```
if values[middle] < query:
    return bFind (values, query, middle+1, stop)
return bFind (values, query, start, middle-1)
```

- d. Your friend suggests that you rather replace the `find` function with the simpler `findAlt` function. Is this a good idea? *Justify your answer.* [2]

No.[1] FindAlt is a linear search and will be must less efficient that a binary search for a list that is already sorted.[1]

Question 5 [10]

Examine the **Q5.py module** listed at the end of the exam paper and answer the following questions.

a. You are asked to test the `Myst(n)` function using a range of testing strategies.

a) Does the function call

`Myst(6)`

constitute **statement coverage** testing of function `Myst`? *Explain your answer.*

[2]

Yes. [1] All lines are executed[1]

b) Write down a **minimal** set of inputs to this function that would constitute a complete **path test**.

[1]

Two calls are required, one with $n \leq 0$, one with $n > 0$ e.g.

`Myst(0)`

`Myst(6)`

b. Write down the exact output when the **Q5.py** module is executed.

[1]

[1, 1, 2, 3, 5, 8] #one small error = 1 mark, else 0

c. Rewrite the `Myst(n)` function below to use recursion instead of iteration.

[4]

`def MystRec(n):`

```

def MystRec(n): #no recursion, no marks!
    if n > 0: #stopping case 1
        if n == 1: return [1] #stopping case 2
        if n == 2: return [1,1] #[2] stopping cases
        tmp=MystRec(n-1) #[1] recursive call
        tmp.append(tmp[-1]+tmp[-2]) #[1] appending correctly
        return tmp #
    return []

```

- d. Calculate $16_{10} - 5_{10}$ using **8-bit 1's complement** binary addition. Show **all** calculations. [2]

```

610-1510
= 0001 0000 + 1's complement (0000 0101)
=0001 0000 + 1111 1010 [1 mark for correct sum showing 1's complement of 15]
= addition showing carry into leftmost bit to give 0000 1011 [1]
must show 8 bits!

```

Question 6 [10]

Many people are saying that modern politicians are using increasingly simplistic language in an effort to appeal to a wider audience. You have decided to investigate this by analysing the frequency of word lengths that they use. You have started with the code listed below.

```
#module finalQ.py

def analyse_text(filename):
    #you must write this code

def barChart(valuesDict):
    total=0
    for v in valuesDict: total+=valuesDict[v]
    row="{:>2} | {:<10}"
    c='*'
    barMax=40
    for v in sorted(valuesDict):
        print(row.format(v,c*(barMax*valuesDict[v]//total)))

x=analyse_text("trump.txt")
barChart(x)
```

This program will calculate the word length frequency in a file of a speech and then output a histogram of word length and frequency. The program should ignore all punctuation such as ". , ; ' ?"; e.g "don't" will be counted as 4-letter word. For example, if "trump.txt" contains the following lines:

And they're making him do it because he's crashing in the polls.
So I don't know.

I just don't know. He's crashing.

the output would be:

```
1 | ***
2 | *****
3 | *****
4 | *****
5 | *
6 | ***
7 | *
8 | ***
```

Write the `analyse_text` function on the following page.

[illegible]

```
# module: finalQ.py

def analyse_text(filename):
```



```

#you must write this code
###remove below from EXAM!!!!
f=open(filename,'r') #readin correctly from file [2]
data=f.readlines()
f.close()
wordDict={} #creating a dictionary [1]
for line in data: #correct nested loops [1]
    words=line.split()
    for w in words:
        w=w.strip(",!?.\n") #removing punctuation outside [1]
        w=w.replace("'", "") #removing punctuation inside [1]
        l=len(w) #a dictionary which calculates the length [3]
        if l in wordDict:
            wordDict[l]+=1
        else: wordDict[l]=1
    return wordDict #returning dictionary [1]
### End remove from exam

def barChart(valuesDict):
    total=0
    for v in valuesDict: total+=valuesDict[v]
    row="{:>2} | {:<10}"
    c='*'
    barMax=40
    for v in sorted(valuesDict):
        print(row.format(v,c*(barMax*valuesDict[v]//total)))

x=analyse_text("trump.txt")

barChart(x)

```

Code examples for the examination (you may detach these sheets).

Question 4

```
#Module Q4.py

def extract(filename):
    f=open(filename,'r')
    x=[]
    for line in f:
        line=line.strip('\n')
        line=line.split(',')
        x.append(line)
    f.close()
    return x

def locate(fName1,fName2,arr2D):
    f=open(fName1,'r')
    f2=open(fName2,'w')
    for i in range(len(arr2D)):
        arr2D[i].sort()
        query=f.readline()
        query=query.strip('\n')
        location=find(query,arr2D[i])
        print(query,location,file=f2)
    f.close()
    f2.close()

def find(query,values):
    return bFind(values, query,0,len(values)-1)

def bFind(values, query, start, stop):
    """Binary search"""
    if start > stop:
        return -1
    middle = (start + stop) // 2
    if values[middle] == query:
        return middle
#Code lines here hidden
```

```

def findAlt(val,arr):
    for i in range(len(arr)):
        if arr[i]==val:
            return i
    return -1

data=extract("happy.txt")
print(data)
locate("values.txt","exam.txt",data)
print(data)

```

Files in the same directory contain the following lines of text:

```

happy.txt
    80,30,20
    c,d,f,a,b
    10,3,2,7,1

```

```

values.txt
    20
    c
    50

```

```

exam.txt
    Fido,dog,3
    Mortimer,mouse,1

```

Question 5

```
#Module Q5.py
```

```

def Myst(n):
    output=[]
    a,b=1,1
    for i in range(n):
        output.append(a)
        a,b=b,a+b
    return output

print(Myst(6))

```