University of Cape Town ~~ Department of Computer Science

Computer Science 1015F ~~ 2015

Class Test 2 draft

***** Solutions *****

Question 1 - Charles Babbage and the Analytical Engine [20]

In 1837, Charles Babbage designed, but did not build, the mechanical Analytical Engine to be one of the earliest general purpose computers. Programs were written and submitted on sequences of punched cards, in a language that was the equivalent of a modern-day assembly language.

As a programmer, you are fascinated by how Babbage designed his computer and you would like to understand what he did and then simulate the operation of his computer using a Python program.

a) In the context of the original Analytical Engine, what was the hardware and what was the software? Explain your answer. [2]

hardware is the mechanical parts [1] and software is the punched cards [1]

b) Was the programming language used a low level or high level language? Explain your answer. Briefly discuss 2 advantages of such a type of language. [4]

low [1]
it was a form of assembly language, which is closer to what the machine understands [1]
faster programs [1] minimal translation needed [1] can make of all features of machine [1]

c) Besides assemblers, programs can also be translated/processed by compilers and interpreters. What is one major difference between a compiler and an interpreter? [2]

compiler converts program to low level form but interpreter does not convert - it simply executes an equivalent set of low level instructions [2]

d) In our simulation of Babbage's computer, we have the following function to interpret a row of dots on a punched card and return a unique code.

```
def interpret_card ( hole_1, hole_2 ):
    if hole_1==0 and hole_2==0:
        return 0
    elif hole_1==0 and hole_2==1:
        return 1
    elif hole_1==1 and hole_2==0:
        return 2
    elif hole_1==1 and hole_2==1:
        return 3
```

What is the return value if the actual parameters are 1 and 0 (in that order)? [1]

2 [1]

e) What are 2 advantages of using such functions in our programs? [2]

code can be reused [1]
more modularity so easier to write long programs [1]

easier to test short bits at a time [1]
...

f) Where would a docstring be inserted in the `interpret_card` function? [1]

immediately after the function header / first line [1]

g) Assuming there are no errors in the parameters and these are all the possible and valid values, write a single line version of this function that is both shorter and more computationally efficient. [1]

*return hole_1*2 + hole_2 [1]*

h) In our simulation, we also have the following function to perform computation on a globally-defined variable based on an operation code and integer value.

```
def compute ( operation, val ):
    # missing line
    if operation == 0:
        answer += val
    elif operation == 1:
        answer -= val
    elif operation == 2:
        answer *= val
```

What is the missing line of code to make this function work without an error? [1]

global answer [1]

i) Changing global variables within functions is arguably a bad idea. Explain how you could rewrite this function so it does not change a global variable. [2]

Pass the answer in as a parameter and return the modified answer [2]

j) The following poorly-designed program extracts each integer digit within a string and prints it to the screen. Reorganize/rewrite this program so that: it defines a function `get_digit (string, position)` to return the integer digit at a specified position in a string; the function is used within a `main` function that produces the same output as the original program; and the program includes the special Python code that allows it to be used both as a module and a standalone program. [4]

```
test_number = '0110'
digit = test_number[0]
print (ord (digit) - ord ('0'))
digit = test_number[1]
print (ord (digit) - ord ('0'))
digit = test_number[2]
print (ord (digit) - ord ('0'))
```

```
   digit = test_number[3]
   print (ord (digit) - ord ('0'))
```

```
def get_digit( string, position ): [1]
   digit = string[position]
   return ord (digit) - ord ('0') [1]

def main ():
   test_number = '0110'
   for i in range (len (test_number)):
      print (get_digit(test_number, i)) [1]

if __name__=="__main__": [1]
   main ()
```

Question 2 – Arrays and Testing [20]

Examine the `test2_2016.py` module listed on the last sheet of the test and answer the following questions.

a) Write down the **exact output** when this module is executed (e.g. when the user presses the "Run" button in Wing101). [4]

[2, 4] #[1] mark
[4] #[1] mark
['d', 'o', 't'] #[1] mark
[] #[1] mark *[4]*

b) You have been asked to test the function `daFunc()` 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 pairs of lists. *[2]*

c) Does the function call

```
daFunc([2,3],[1,4])
```

provide **statement coverage** testing of function `daFunc`? Explain your answer. [2]

Yes. [1] All lines in the programme are translated and executed.[1]

d) Does the function call

```
daFunc([2,3],[1,4])
```

provide **path** testing of function `daFunc`? Explain your answer. [2]

No. [1] An empty list for x will result in the loop not executing [1] [or another valid alternative path]

e) Each of the modules below will generate an error. In each case, explain **both** what **kind** of error it is and how to **fix** it. [5]

A.

```
import Test2_2016
print(Test2_2016.daFunc([2 3 4],[2 5 6]))
```

Syntax error (or compile-time error) [1] – fix by putting commas between the list elements.[1]

B.

```
import Test2_2016
print(Test2_2016.daFunc([2,3],[3]))
```

Runtime error (or list indexing error) [1] – fix by checking that lists x and y are of equal length (or y is not shorter than x) before entering loop [2] other valid fix OK too.

f) Rewrite the code for the function `daFunc(x,y)` so that it works as follows. This function should return a new list containing all the elements of list `x` that also occur in list `y`. For example, in the Python3 interpreter:

```
>>>daFunc([2,3,5],[5,3])
    [3, 5]
>>>daFunc([2,3,5],[6])
    []
>>>daFunc(['a','b','c'],['c','b','a'])
    ['a', 'b', 'c']
```

Complete the code below: [5]

```
def daFunc(x,y):
```

One correct answer is:

```
def daFunc2(x,y):
  z=[]  #[1] mark
  for i in x:  #[1] mark
     for j in y: #[1] mark
        if i==j: #[1] mark
           z.append(i) #[1] mark
  return z # this must be here - [-1] if not.
```