Please fill in your Student Number and Name.

Student Number : ________________________________

Name:

____

____

Student Number:

____

University of Cape Town ~ Department of Computer Science

Computer Science 1018F ~ 2008

June Exam

Question	Max	Internal	External	Question	Max	Internal	External
1	10			7	10		
2	14			8	5		
3	6			9	10		
4	12						
5	8						
6	25						
				TOTAL	**100**		

Marks : 100

Time : 180 minutes

Instructions:

a) Answer all questions.

b) Write your answers in the space provided.

c) Show all calculations where applicable.

Question 1 [10]

a) Given a dictionary declared as: **`actors = {'John':'Cleese'},`** why is **`actors['John']`** a valid operation but not **`actors['Cleese']`**? [2]

A dictionary uses keys to retrieve values. Here 'John' is a key used to retrieve the associated value 'Cleese'. However, values cannot be used to retrieve 'keys'

b) Python implements automatic garbage collection. What does this mean and what implications does it have for programming in Python? [2]

Automatic garbage collection means that memory occupied by objects is deleted automatically once the object goes out of scope. This implies that a programmer does not have to explicitly delete objects as is required by Java and Python.

c) What is the difference between private and public methods and attributes of a class? How are these declared in Python? [2]

Public methods and attributes are accessible outside the class, while private methods and attributes can only be accessed by methods within the class. By default attributes and methods are public, the "__" prefix is used to make them private.

d) Rewrite the following Python statement using a more conventional control structure: [2]

```
r = (a > b and [a] or [b])[0]
```

```
if a > b:    [1/2]
   r = a     [1/2]
else:        [1/2
   r = b     [1/2]
```

e) The Python string is equivalent to another native datatype. What is this datatype and what implications does this have for operations on strings? [2]

A tuple of characters. This is an imutable datatype so strings cannot be modified (by extend, append, insert, for instance) once created.

Question 2 [14]

Write a function called *Telegram* that reads in a text file containing an English message and converts it into a format suitable for telegram transmission. This means that all text must be capitalised and punctuation marks are converted to equivalent words (e.g., "." becomes "STOP" and "," becomes "COMMA"). You may assume that full stops and commas are the only punctuation. Finally, all spaces are replaced by full stops. For example, *Telegram* will convert the message "Hit Iceberg. Sinking, send rescue." to:

"HIT.ICEBERG.STOP.SINKING.COMMA.SEND.RESCUE.STOP".

Telegram should take the name of the message file as an argument. If a filename is not supplied the function should try to open "message.txt" in the current working directory and it should report any errors encountered in opening the file. The reformatted message should be appended to the end of the file containing the original message.

Hint: The *string* module has a function, *upper(s)*, which converts a string *s* to upper case.

```
import string                                                              [1/2]


def Telegram(filename = "message.txt"):                                    [1/2]
  """Convert the text in <filename> to telegram formatting and append
  to the end of the file."""                                               [1/2]


  try:                                                                     [1/2]
    f = open(filename, 'r')                                                [1/2]
  except IOError:                                                          [1/2]
    print 'unable to open file'                                            [1/2]
    return False


  buf = f.read()                                                           [1/2]
  words = buf.split(' ')                                                   [1/2]
  tel = “”                                                                 [1/2]
  for w in words:                                                          [1/2]
    for l in w:                                                            [1/2]
      if l == '.':                                                         [1/2]
        tel = tel + '.STOP'                                                [1/2]
      elif l == ',':                                                       [1/2]
```

```
            tel = tel + '.COMMA'                                   [1/2]
        else:                                                      [1/2]
            tel = tel + string.upper(l)                            [1]
    tel = tel + '.'                                                [1/2]
tel = tel[:-1] # one too many '.'                                  [1/2]
f.close()                                                          [1/2]


try:                                                               [1/2]
    f = open(filename, 'a')                                        [1/2]
except IOError:                                                    [1/2]
    print 'unable to write to file'                                [1/2]
    return False
f.write(tel)                                                       [1/2]
f.close()                                                          [1/2]
return True
```

Question 3 [6]

You are provided with the following lists of American Civil War generals:

```
North = ['Grant', 'Sheridan', 'Hood']
South = ['Lee', 'Jackson']
```

Write out the contents of these lists after each of the following operations are applied in sequence:

a) **`South.insert(len(South), North.pop())`** [2]

North = [‘Grant’, ‘Sheridan’], South = [‘Lee’, ‘Jackson’, ‘Hood’]

b) **`North.append(North[:1])`**
`South = South[:]` [2]

North = [‘Grant’, ‘Sheridan’, [‘Grant’]], South = [‘Lee’, ‘Jackson’, ‘Hood’]

c) **`North.remove(['Grant'])`**
`North.extend(South)`
`South = []` [2]

North = [‘Grant’, ‘Sheridan’, ‘Lee’, ‘Jackson’, ‘Hood’], South = []

Question 4 [12]

a) The puzzle game Sudoku requires that every row (and column) contains the numbers 1, 2, …, 9 with no repetition but in any order. Write Python code to read in a set of single digit integers separated by spaces from the keyboard using raw_input, determine if they form a permutation of the numbers 1-9 and print out 'Correct' if they do, or 'Incorrect' if they do not. In either case you should also print out the sum of the numbers entered. [8]

```
s = raw_input('Enter a row of numbers:')                [1/2]
chars = s.split(' ')                                    [1/2]
nums = []                                               [1/2]
for el in chars:                                        [1/2]
    nums.append(int(el))                                [1]
sum = 0                                                 [1/2]
for n in nums:                                          [1/2]
  sum += n                                              [1/2]
for i in range(9):                                      [1/2]
    nums.remove(i+1)                                    [1/2]
if nums == []:                                          [1/2]
    print 'Correct'                                     [1/2]
else:                                                   [1/2]
    print 'Incorrect'                                   [1/2]
print sum                                               [1/2]
```

b) Python is classified as a language with strong and dynamic typing. Give examples from your answer to part a) which demonstrate this and explain why they do so? [4]

nums.append(int(el)) – the char types entered using raw input must be explicitly converted to integer, demonstrating strong typing *[2]*

sum = 0 – the type of sum does not need to be explicitly declared, demonstrating dynamic typing *[2]*

Question 5 [8]

You are provided with the following Python classes:

```
class Mythical:
    def __init__(self):
        self.magical = True

class Carnivore:
    trophies = 0

    def hunt(self):
         Carnivore.trophies += 1

class Jabberwock(Mythical, Carnivore):
    def hunt(self):
        Carnivore.hunt(self)
        print 'The jaws that bite, the claws that catch!'

    def __vorpal(self):
        print 'The vorpal blade went snicker-snack!'
```

Write down the Python output of the following code. Explain your answers in each instance.

a) `print Jabberwock.trophies` [2]

output: 0

trophies is a class attribute of Carnivore and by inheritance Jabberwock. It is a single global value common to all Jabberwock objects and the class itself.

b)
```
jay = Jabberwock()
jay.hunt()
print jay.trophies
```
[2]

output: The jaws that bite, the claws that catch

1

the hunt method calls the parent class of Jabberwock which increments the trophies counter

c) `print jay.magical` [2]

output: True

magical is a public attribute of mythical initialised during creation of a Jabberwock object

d) **`jay.__vorpal()`** [2]

output: AttributeError

vorpal is private and cannot be called outside of the Jabberwock class

Question 6 [25]

You are provided with the following Python code:

```
"""Unit test for 12-24 hour clock conversions"""
import Clock
import unittest

class SuccessCheck(unittest.TestCase):
   knownValues = (( '12:00am', '0h00'),
          ('8:00am', '8h00'),
          ('3:15pm', '15h15'),
          ('10:40pm', '22h40'))

   def testKnown24(self):
      for timeIn,timeOut in self.knownValues:
         result = Clock.Time24(timeIn)
         self.assertEqual(result, timeOut)

   def testKnown12(self):
      for timeOut,timeIn in self.knownValues:
         result = Clock.Time12(timeIn)
         self.assertEqual(result, timeOut)

   def testSanity(self):
      """ code deleted """

class FailureCheck(unittest.TestCase):
   def testFormat12(self):
      self.assertRaises(Clock.invalidFormat,Clock.Time24,"9:00a ")

   def testFormat24(self):
      self.assertRaises(Clock.invalidFormat,Clock.Time12,"9 h 34")

   def testIncorrectRange12(self):
      self.assertRaises(Clock.outOfRange,Clock.Time24,"13:24pm")

   def testIncorrectRange24(self):
      self.assertRaises(Clock.outOfRange,Clock.Time12,"24h24")

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

a) Unit testing is an example of black box testing. What is **black box** testing? [2]

This testing method treats program as a "black box" – i.e. only consider the inputs and outputs[1] and don't look at the code.[1] The tests are based on the specifications - Base test data on specifications: what the program is supposed to do

b) The class **`SuccessCheck`** does not take the principles of equivalence testing and boundary analysis into account.

i. What is boundary analysis and why is it necessary? [2]

Boundary analysis involves looking at the values on either =side of an equivalence class. This is where most of the errors occur.

ii. Write down a new list of values for **`knownValues`** that represents an appropriate set of equivalence classes and boundary values for this problem. [3]

```
knownValues = (( '11:59pm', '23h59'),( '12:00am', '0h00'),( '12:01am', '0h01'),
     ('8:00am', '8h00'), ( '11:59am', '11h59'),( '12:00pm', '12h00'),( '12:01pm', '12h01'),
     ('3:15pm', '15h15'))
```

c) Tests are typically divided into tests for success, tests for failure and other tests, such as tests for sanity.

i. What is a test for failure? [1]

A test that checks that the unit raises the correct Exceptions with incorrect input

ii. The **`testSanity`** method should run a sanity check. What is a *sanity check*? [1]

Given a function f(x) and it's inverse f'(x), a test that checks whether f'(f(x)) =x

iii. Write down code for the **`testSanity`** method. This should run a sanity check for all the values listed in **`knownValues`**. [4]

```
def testSanity(self):
     for timeIn,timeOut in self.knownValues:
         result = Clock.Time24(timeIn)
         self.assertEqual(timeIn, Clock.Time12(result))
```

d) When running the unittest module, both failures and errors can occur. Explain clearly what the difference between an error and a failure is. [2]

A failure is a call to an assertXYZ method, like assertEqual or assertRaises, that fails because the asserted condition is not true or the expected exception was not raised- failures are caused by incorrect results. An error is any other sort of exception raised in the code you're testing or the unit test case itself. Errors are caused by incorrect code

e) Given the skeleton below for **`Clock.py`**, complete the **`Time24`** method so that it will pass the unit test listed above. [10]

```
import re
import sys

#define exceptions
class invalidFormat(Exception): pass
class outOfRange(Exception):pass

def Time12(string):
    """ convert from 24 hour to 12 hour clock """

def Time24(string):
    """ convert from 12 hour to 24 hour clock """
```

There are a number of solutions to this!

```
def Time24(string):
  """ """
  RE=r'\b(\d?\d):(\d\d)([ap]m)\b' [2]
  grps=re.match(RE,string) [1]
  if (not grps):
    raise invalidFormat [1]
    return
  h = int(grps.group(1))
  if not (1 <=h <=12): [1]
    raise outOfRange [1]
    return
  if((grps.group(3)=="pm")): [1]
    if (1<h<12): h=h+12 [1]
  else:
    if(h==12): h=0 [1]
```

```
result=  str(h) + 'h'+ grps.group(2)  [1]

return result
```

Question 7 [10]

Examine the python code below:

```
startText = """But four young Oysters hurried up,
All eager for the treat:
Their coats were brushed, their faces washed,
Their shoes were clean and neat--"""

repStr = '*'
print re.sub(regExp, repStr, startText, re.MULTILINE)
```

In each question that follows, indicate the value(s) of **`regExp`** that will print the displayed text.

a)

```
But four young Oysters hurried up,
All *ger for the tr*t:
Their coats were brushed, their faces washed,
Their shoes were cl*n and n*t--
```

A. r'[ea]'
B. r'ea'
C. r'\bea'
D. B and C

B

b)

```
But four young Oysters hurried up,
All eager for the tr*t:
Their coats were brushed, their faces washed,
Their shoes were cl*n and n*t--
```

A. r'\Bea'
B. r'ea'
C. r'\bea'
D. A and B

A

c)

```
But four young Oysters hurried up,
All eager for the *:
Their coats were brushed, their faces washed,
Their shoes were clean and *--
```

A. r'[a-z]*eat'
B. r'(n|t|r)*eat'
C. r'\b[a-z]{1,2}eat'
D. All of the above

D

d)

```
But f*r y*ng Oysters hurr*d up,
All *ger for the tr*t:
Th*r c*ts were brushed, th*r faces washed,
Th*r sh*s were cl*n and n*t--
```

A. r'[aeiou][aeiou]'
B. r'[aeiou].'
C. r'[aeiou][aeiou]*'
D. A and C

A

e)

```
But four young Oysters hurried up,
All eager for the treat:
Their coats w* brushed, their faces washed,
Their shoes w* clean and neat--
```

A. r'ere'
B. r'\bere'
C. r'([aeiou])[a-z]\1'
D. A and C

D

Question 8 [5]

a) Write down a regular expression that will match land-line telephone numbers in either of the formats “(021) 6505107” or “+27 21 6505107”.

The first format starts with regional dialing code in parentheses, comprising a zero followed by two or three digits. A space separates this from the local dialling code, comprising 6 or 7 digits, with no spaces.

The second format lists the international prefix first – a ‘+’ symbol, followed by the country code (between 1 and 3 digits, e.g. “+27” for South Africa or “+1” for the U.S.A.), a space and then the regional dialling code (no parenthesis).

Your expression should match strings such as “(0532) 6565656” and “+27 11 999999”, but not to strings such as “0532 6565656” and “(021)123123”. [5]

```
r’((\(0\d{2,3}\) )|(\+\d{1,3} \d{2,3})) \d{6,7}
```

Question 9: Number Systems [10]

a) Write down the octal representation of the binary number 1111010_2. Show all your working.[2]

172_8

b) Write down the binary representation of the hexadecimal number CAB_{16} . Show all your working. [2]

110010101011_2

c) Write down the binary representation of the decimal number 43.375_{10}. Show all your working. [3]

101011.011_2

d) What is the value of the floating point number below? Assume IEEE754 single precision format, i.e., the left-most bit is the sign bit, the next 8 bits are the biased exponent, and the right-most 23 bits are the significand. Show all your working. [3]