Name:

____

____

Student Number:

____

Please fill in your Student Number and Name.

Student Number : __________________________________

University of Cape Town ~ Department of Computer Science

Computer Science 1015F ~ 2012

Test 2

***** SOLUTIONS *****

Question	**Max**	**Mark**	**Marker**
1	6		
2	8		
3	8		
3	8		
TOTAL	**30**		

Marks : 30

Time : 40 minutes

Instructions:

a) Answer all questions.

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

c) Show all calculations where applicable.

Question 1 [6]

The function `categorize(age)` below takes a person's age (in years) as a parameter and **returns** an age category in the range “child”, “teenager”, “adult” and “pensioner”. For example, the statement

```
print(categorize(6))
```

will display:

```
child
```

whereas

```
print(categorize(67))
```

will display:

```
pensioner
```

Complete the code for this function, below. [6]

```
def categorize(age):
```

```
def categorize(age):
  if age <0:  #[1]
    return("Error")
  elif age<=12:
    return("Teenager") #[1]
  elif age<=65:
    return("adult") #[1]
  elif age<=125:
    return("Pensioner") #[1]
  else:
    return("error")  #[1]
and one for “return”, not “print”
```

Question 2 [8]

Examine the module listed below:

```
def twist(input):
   max=len(input)
   for i in range(max):
      print(input)
      input= input[-1]+input[0:max-1]

twist("Neo")
```

(a) Explain, in general and in clear English, what the function "twist" does. [2]

Prints the string as a box [1], shifting the input string one character along with each successive line[1]

(b) Write down the **exact output** of the module listed above when it is run in the Python interpreter (i.e. when you press "Run" in the Wing IDE). [3]

```
Neo
oNe
eoN
```

(c) The function `twist` uses a `for` (definite) loop. Rewrite the function in the space below so that it **works exactly the same as before** but now uses a `while` (indefinite) loop. [3]

```
def twist(input):
```

```
#QTest2.py
def twist(input):
  end=len(input)
  i=end  #[1]
  while i>0: #[1]
    print(input)
    input= input[-1]+input[0:end-1]
    i-=1  #[1]
#or equivalnet working code with while loop
```

Question 3 [8]

Examine the module listed below:

```
def mystery(word):
   output=''
   for j in word.lower():
      if j=='k' or j=='e' or j=='a' or j=='n' or j=='u':
         print("--disinfecting")
         break
      output+=j
   else:
      print("--clean")
   return output

x=mystery("Morpheus")
print(x)
y=mystery(x)
print(y)
```

(a) Explain clearly and concisely why this line in the mystery function:

```
if j=='k' or j=='e' or j=='a' or j=='n' or j=='u':
```

cannot be replaced with this line:

```
if j== 'k' or 'e' or 'a' or 'n' or u':
```

[2]

The strings 'k', 'e', 'a' etc. all evaluate as true, so this expression will always be true.

(b) Write down the **exact output** of this module when it is run in the Python interpreter (i.e. when you press “Run” in the Wing IDE). [6]

--disinfecting

morph

--clean

morph

Question 4 [8]

Answer the following questions based on the program below.

```
n = eval (input ("Enter a value:"))

left = 1
right = 1
guess = 1
while not left <= n <= right:
   length = right-left+1
   left = right+1
   right = left+length
   guess += 1

print (guess)
```

(a) What is an equivalence class? [1]

set of input values for which program behaves in the same manner

(b) Explain, using examples, what equivalence classes can be used to test this program. [2]

sets of sequential integers of increasing size from 1 onwards [1,1], [2,3], [4,6], [7,10]...

(c) Which values in the range 0-10 are NOT candidate boundary values? [1]

none! all are boundary values

(d) For what value in the range 0-10 will this program not function correctly? Is this a syntax error or a logic error? Explain your answer. [3]

0 [1], logic error [0], program still has correct structure and executes but produces incorrect output [2]

(e) What testing technique is **better** than equivalence classes and boundary values? Explain your answer. [1]

Exhaustive testing, because it checks every possible input [1]

or

Path testing or statement coverage, because it is faster [1]