Name:

____

____

Student Number:

____

Please fill in your Student Number and Name.

Student Number : __________________________________

University of Cape Town ~ Department of Computer Science

Computer Science 1018F ~ 2009

Test 2

Question	**Max**	**Mark**	**Internal**	**External**
1	3			
2	15			
3	12			
TOTAL	**30**			

Marks : 30

Time : 40 minutes

Instructions:

a) Answer all questions.

b) Write your answers, in pen, in the space provided.

c) Show all calculations where applicable.

Question 1 [3]

Given a list: `fruit = ['persimmon', 'granadilla', 'litchi']`

What would the *newfruit* list look like after each of the following operations. [3]

a)

```
newfruit = fruit[:-1]
```

newfruit = [‘persimmon’, ‘granadilla’]

b)

```
newfruit = [f*2 for f in fruit]
```

newfruit = [‘persimmonpersimmon’, ‘granadillagranadilla’, ‘litchilitchi’]

c)

```
newfruit = fruit
newfruit.append(['kiwi', 'plantain'])
```

newfruit = [‘persimmon’, ‘granadilla’, ‘litchi’, [‘kiwi’, ‘plantain’]]

Question 2 [15]

a) Your task is to create a telephone directory using a dictionary data structure that will allow a programmer to look up a telephone number given a person's name. This question has two parts:

   i. Write a method *telparse(telfile)* that opens a file called telfile containing a list of names and associated telephone numbers. Each name and number is linked by a '=' (i.e., name1=telnum1) and telephone entries are seperated by spaces. A typical file might have the form:

      James=6504058 Hussein=6505106

      The *telparse* method should return a list of strings of the form `['name1=telnum1', 'name2=telnum2', …]`. Be sure to use proper exception handling when opening the *telfile* and close any open files once finished. [6]

```
"""Parse the telfile into a list of name and telephone
number pairs."""                                        [1/2]

try:                                                    [1/2]
    tf = open(telfile, 'r')                             [1/2]
    ttxt = tf.read()                                    [1/2]
    tf.close()                                          [1/2]
except IOError:                                         [1/2]
    print 'Error in File IO'                            [1/2]
    return []                                           [1/2]
tlist = ttxt.split(' ')                                  [1]
return tlist                                            [1/2]
```

ii. Write a method *teldir(tellist)* that takes the list of names and telephone number pairs from part (i) and creates and returns a dictionary indexed by name. Hint: you will need to use the ‘=’ to split apart the name and telephone number. [5]

```
def teldir(tellist):                                              [1/2]
  """Return a telephone dictionary from a list of names and
  telephone numbers."""                                           [1/2]

  tdict = {}                                                      [1/2]
  for telentry in tellist:                                          [1]
     tel = telentry.split(‘=’)                                      [1]
     tdict[tel[0]] = tel[1]                                         [1]
return tdict                                                      [1/2]
```

b) Mention two (2) practices of effective defensive programming and explain why they lead to fewer bugs or make debugging easier in the resulting programs. [4]

Plan before coding [1] – solution is propertly structured and hence easier to debug [1]

Comment while coding [1] – means that you (or another coder) can understand the code when you return to it later [1]

Use pre- and post-conditions [1] – catches bugs that arise from incorrect input or output values [1]

Catch exceptions [1] – makes sure that you identify and helpfully label bugs at the point where they occur [1]

Test [1] - the only way to find a range of bugs before the software is delivered [1]

Any two accepted. Arguments don’t have to match exactly but must be plausible.

Question 3 [12]

Visual Python has a class which supports a variety of operations on vectors. You are given the following class declaration:

```
Class Vector():
    """A mathematical entity with three elements i, j, k.
    It represents direction and magnitude in a three-
    dimensional coordinate system."""
```

a) Create the following methods for your version of the Vector class:

   i. An initialisation method that assigns its three parameters to the i, j, k instance variables. The parameters are optional and default to a value of 0.0.

   ii. An add method that takes a vector as argument and performs component-wise addition with the current vector and returns the result as a new vector. Called as: v = v1.add(v2)

   iii. A length method that returns the magnitude of the current vector (the square root of the sum of the squares of the vector elements). Called as: l = v.length() [10]

```
def __init__(self, a = 0.0, b = 0.0, c = 0.0):                        [1]
   "initiliase by assigning (a,b,c) to the vector"                    [1/2]
   self.i = a                                                         [1/2]
   self.j = b                                                         [1/2]
   self.k = c                                                         [1/2]
def add(self, Vector v2):                                              [1]
   "add v2 to the current vector and return the sum"                  [1/2]
   v = Vector(self.i+v2.i, self.j+v2.j, self.k+v2.k)                [1 1/2]
   return v                                                           [1/2]
def length(self):                                                      [1]
   "return the length of the vector"                                  [1/2]
   return sqrt(self.i*self.i + self.j*self.j + self.k*self.k)          [2]
```

b) It might be easier for a user of the vector class to use the '+' addition operator (e.g., `v = v1 + v2`) instead of the add method (e.g., `v = v1.add(v2)`). How could this be done and what is this process called? [2]

By using a special class method (__add__). [1] It is called operator overloading or polymorphism [1]