Final

Bioinformatics

Sup'Biotech 3

Python

Pierre Parutto

November 29, 2015

Firstname: ______________________________

Lastname: ______________________________

Allowed Documents: No document; No calculator.

Answer directly on the subject inside the provided boxes. If you need more space there is one blank page at the end.

1 Cours (8 points)

1.1 QCM (5 points)

Multiple answers can be correct; For each question: all good answers: **+0.5 point**; a bad answer or missing good answers: **-0.25 point**; No answer: **0 point**. If the total grade of the mcq is < 0 the grade is set to 0.

Questions	Answers
1. A recursive function is	☐ a function that has no return value ■ a function that calls itself ☐ a function that calls another function ☐ a function without any loop
2. A recursive function must possess:	■ at least one base case ☐ necessarily multiple base cases ☐ necessarily multiple recursive calls ■ at least one recursive call
3. In a recusrive function, a base case is defined as the case for which:	☐ the function calls itself ☐ the function returns the value `None` ☐ the function returns a call to the function ■ the function returns a value
4. If we try to open in reading mode (mode `"r"`) a file that does not exist then:	■ there is an error ☐ the file is created ☐ another similar file is opened ☐ the file is opened in writing mode instead
5. One can directly read the whole content of a file with:	☐ the function `readoneline` ■ the function `readlines` ☐ A `while` loop ■ A `for` loop
6. What is (are) the function(s) that allow(s) to create matrices ?	☐ `matrix` ■ `array` ■ `zeros` ☐ `twos`
7. Let `M` be a variable of type `array` (numpy matrix), then `M[i,:]` :	☐ accesses to one element ■ accesses to a line ☐ accesses to a column ☐ produces an error
8. Let `M` be a variable of type `array` (numpy matrix) what is (are) the valid proposition(s) :	■ `M.shape` ☐ `M.allelements()` ■ `M.size` ■ `M.ndim`
9. The function `range(n)` (n an integer) creates the list:	■ `[0, 1, ..., n - 1]` ☐ `[0, 1, ..., n]` *Continued on the next page...*

Questions	Answers
	☐ `[1, 2, ..., n]` ☐ `[1, 2, ..., n-1]`
10. Let `l` be a list, in `for e in l`, the variable `e` successively takes the values:	☐ the positions of each elements in `l` ☐ the value of the first element of `l` [✓ the value of each elements of `l`] ☐ there is an error

1.2 Questions (3 points)

1.3 Code Blocs (1 point)

Represent all the blocs of codes by vertical lines at the left of the following code:

```
def toto(a):
        b = a * a / a
        return b

coucou = 5
for k in [1,2,3]
        k = k + 2
        if k != coucou:
                print(3)
        elif k == coucou * 5
                print(8)
                if k == 10:
                        print(50010)
                        k = k +1
                print(toto(k))
```

Correction:

```
│def toto(a):
│ │b = a * a / a
│ │return b
│
│coucou = 5
│for k in [1,2,3]
│ │k = k + 2
│ │if k != coucou:
│ │ │print(3)
│ │elif k == coucou * 5
│ │ │print(8)
│ │ │if k == 10:
│ │ │ │print(50010)
│ │ │ │k = k +1
│ │ │print(toto(k))
```

1.4 Displaying (1 point)

Consider the code:

```
def aplusb(a, b):
        return a+b
def atimesb(a,b):
        print a*b
aplusb(5,3)
print(atimesb(3,5))
```

1. What does such code print ?
2. Modify it so that it exactly prints:

```
8
15
```

Correction:
As is, it prints:

```
15
None
```

```
def aplusb(a, b):
        return a+b
def atimesb(a,b):
        print a*b
print(aplusb(5,3))
atimesb(3,5)
```

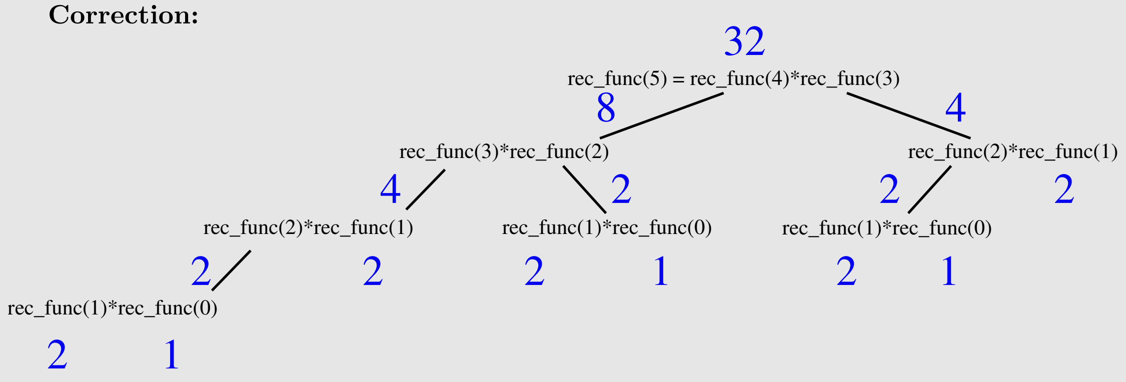
1.5 Recursive Calls (1 point)

Consider the following recursive function:

```
def rec_func(n):
        if n == 0 or n == 1:
                return n + 1
        return rec_func(n-1) * rec_func(n-2)
```

Write the list of all the recursive calls done when evaluating de `rec_func(5)` and give the result of this evaluation.

Correction:



2 Python (5 points)

1 point for each correct function.

2.1 Sequence

Consider the following sequence u_n:

$$\begin{cases} u_{n+1} & = & 5 * u_n - \frac{u_n}{2} \\ u_0 & = & -0.75 \end{cases}$$

Write a **recursive** function `suite(n: int) -> float` that returns the value u_n.

Example

```
>>> suite(1)
-3.375
>>> suite(2)
-15.1875
```

Correction:

```
def suite(n):
       if n == 0:
              return -0.75
       return 5 * suite(n-1) - suite(n-1) / 2
```

We can also remark that:

$$u_{n+1} = 5 * u_n - \frac{u_n}{2} = u_n(5 - \frac{1}{2}) = 4.5 * u_n$$

Which allows us to write the following function (faster because there is only one recursive call):

```
def suite(n):
       if n == 0:
              return -0.75
       return 4.5 * suite(n-1)
```

2.2 Fichier fasta

A fasta file is defined as:

```
>id1
Seq1
>id2
Seq2
```

Write a function `read_fasta(fname: string) -> list` that returns the list of lists: `[[id1,seq1],...,[idn,seqn]]`. In the returned value, each element is a list that associates the id and the corresponding sequence.

Warning

- The sequences are **always** on one line;
- You must remove the character `">"` at the beginning of the ids (it is not part of the name);

- Do not forget to remove the new lines characters "\n".

Example

Consider the fasta file *1.fasta*

```
>superseq
ATGGT
```

And the file *2.fasta*

```
>s1
AAAAAAA
>s2
GCCG
```

```
>>> read_fasta("1.fasta")
[["superseq", "ATGGT"]]
>>> read_fasta("2.fasta")
[["s1", "AAAAAAA"], ["s2", "GCCG"]]
```

Correction:

```
def read_fasta(fname):
        f = open(fname, 'r')
        res = []
        id = ""
        for line in f:
                if line[0] == ">":
                        id = line[1:].rstrip("\n")
                else:
                        res.append([id, line.rstrip("\n")])
        f.close()
        return res
```

2.3 Recursive File Reading

Write a recursive function `read_file_rec(f: file) -> list` that returns the list of the lines (without the "\n") contained in the file `f`.

Warning

- The file `f` is already opened in reading mode;
- You do not have to close `f`;
- The simplest method is to use the function `readline` that returns the next line of the file or "" if the end of the file is reached;
- Be very careful to call `readline` only once per recursive call.

Example

Consider the file *stairway1.txt*

```
And as we wind on down the road
Our shadows taller than our soul.
```

and the file *stairway2.coucou*

```
There walks a lady we all know
...
```

```
>>> f = open("stairway1.txt", "r")
>>> read_file_rec(f)
["And␣as␣we␣wind␣on␣down␣the␣road", "Our␣shadows␣taller␣than␣our␣soul."]
>>> f.close()
>>> f = open("stairway2.coucou", "r")
>>> read_file_rec(f)
["There␣walks␣a␣lady␣we␣all␣know", "..."]
>>> f.close()
```

Correction:

```
def read_file_rec(f):
        l = f.readline()
        if l == "":
                return []
        return [l.rstrip("\n")] + read_file_rec(f)
```

2.4 Sum Of Elements

Write a function `apluskb(A: array, B: array, k: int) -> array` that returns the matrix sum: `A + k*B`.

Warning

- The matrices `A, B` and the returned matrix have the same size;
- If C is the returned matrix, then you must follow the formula:

$$c_{i,j} = a_{i,j} + k * b_{i,j}$$

where i, j are positions.

Example

Consider the following matrices:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} B = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$$

```
>>> from numpy import *
>>> A = array([[1,2,3],[3,2,1],[1,1,1]])
>>> B = array([[0,1,2],[1,2,3],[2,3,4]])
>>> apluskb(A, B, -1)
array([[1, 1, 1], [2, 0, -2], [-1, -2, -3]])
>>> apluskb(A, B, 2)
array([[1, 4, 7], [5, 6, 7], [5, 7, 9]])
```

Correction:

```
from numpy import *
def apluskb(A, B, k):
        C = zeros(A.shape)

        for i in range(A.shape[0]):
                for j in range(A.shape[1]):
                        C[i,j] = A[i,j] + k * B[i,j]
        return C
```

2.5 Position Of The Minimum

Write a function `mat_argmin(M: array) -> list` that returns the positions of the minimum value in the matrix `M` in the form of a list: `[line, column]`.

Warning

- If the matrix is empty, your function must return the value `None`;
- Think carefuly, you will be penalized if your function needs to go through the matrix more than once;
- Reminder: **to find a minimum**, when the matrix is not empty:
  - The current minimum is the first element of the matrix: `M[0,0]`;
  - We traverse the matrix, each time we find an element smaller than the current minimum, this element becomes the current minimum;
  - At the end of the traversal, we only need to return the value of the current minimum, it is the global minimum of the matrix.

  You need to adapt this algorithm to return the position of this minimum.

Example

Consider the following matrices:

$$A = \begin{pmatrix} 1 & -2 \\ 3 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 15 & 31 & 42 \\ 61 & 72 & 36 \\ 32 & 31 & 4 \end{pmatrix}$$

```
>>> from numpy import *
>>> A = array([[1,-2], [3,2]])
>>> mat_argmin(A)
[0, 1]
>>> B = array([[15,31,42], [61,72,36], [32,31,4]])
>>> mat_argmin(B)
[2, 2]
```

Correction:

```
from numpy import *
def mat_argmin(M):
       if M.size == 0:
              return None
       cur_pos = [0, 0]
       cur_min = M[0, 0]

       for i in range(M.shape[0]):
              for j in range(M.shape[1]):
                     if M[i,j] < cur_min:
                            cur_min = M[i,j]
                            cur_pos = [i,j]
       return cur_pos
```

3 Savoie ou quoi ?! (7 points)

For Non-french speakers[1].

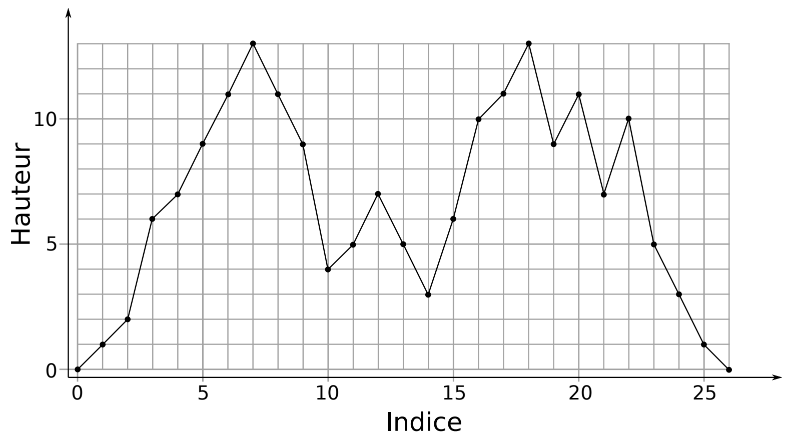
We are in 2050, biotechnologies have lost their swag, you decide to work as the manager of a skii resort.

You are provided with a list of integers representing the height (in arbitrary units) measured every kilometers along a slice of mountain. All along this section we will consider the following example:

```
[0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0]
```

representing the following 2D slice:

[1]For non-french speakers, there is a pun in the title: savoie is a very famous (and nice) region of France with a lot of mountains for skiing; The expression "Savoie ou quoi" comes from a song by Fatal Bazooka: "fou ta cagoule", this expression is pronounced the same way as "ça va ou quoi ?" which can be translated as "what's up?".



3.1 Attention

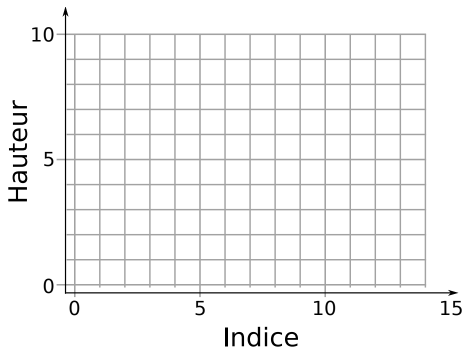
For each question you can use the functions from the previous questions.

3.2 Interpretation (0.5 point)

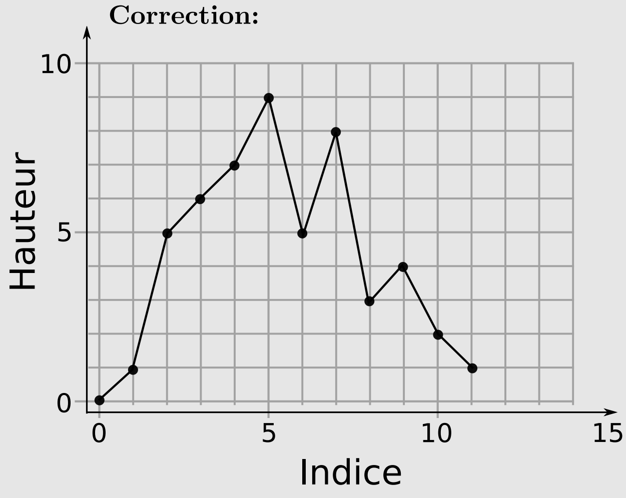
Consider the following list:

```
[0,1,5,6,7,9,5,8,3,4,2,1]
```

Represent the corresponding 2D slice:



Correction:



3.3 Slope (1 point)

The slope has the formula: "height of the next point - height of the current point". Be careful: with a list of n points, we can compute only $n-1$ slopes (we cannot compute the slope of the last point).

Write a function `slope(montagne: list) -> list` that returns the list containing the slopes for each point of the list `montagne` except for the last one.

Example

```
>>> slope([0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0])
[1,1,4,1,2,2,2,-2,-2,-5,1,2,-2,-2,3,4,1,2,-4,2,-4,3,-5,-2,-2,-1]
```

Correction:

```
def slope(l):
        res = []
        for i in range(1, len(l)):
                res.append(l[i]-l[i-1])
        return res
```

3.4 Mounts And Valleys (0.5 points)

Give a method that allows to determine a summit and another method to determine a valley (its lowest point)

Correction:

- Summit: shift from positive to negative slope;
- Valley : shift from negative to positive slope.

3.5 Finding The Summits (1 point)

Write a function `find_summit(montagne: list) -> list` that returns the list of the positions of all the summits in the list `montagne`.

Example

```
>>> find_summit([0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0])
[7, 12, 18, 20, 22]
```

Correction:

```
def find_summit(montagne):
        res = []
        slopes = slope(montagne)

        for i in range(len(slopes) - 1):
                if slopes[i] > 0 and slopes[i+1] < 0:
                        res.append(i+1)
        return res
```

3.6 Finding The Valleys (1 point)

Write a function `find_valley(montagne: list) -> list` that returns the list of the positions of all the valleys in the list `montagne`.

Example

```
>>> find_valley([0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0])
[10, 14, 19, 21]
```

Correction:

```
def find_valley(montagne):
        res = []
        slopes = slope(montagne)

        for i in range(len(slopes) - 1):
                if slopes[i] < 0 and slopes[i+1] > 0:
                        res.append(i+1)
        return res
```

3.7 Skii Track (2 points)

We are going to label the different skii tracks in our mountain. The conditions for labeling a track are the following:

1. A track can only be at a height > 3 and < 12 (both non-included) and must have a slope $\neq 0$;
2. The difficulty is assigned as:
   - green : slope is 1 or -1;
   - blue : slope is 2 or -2;
   - red : slope is 3 or -3;
   - black : slope is > 3 or < -3.

Write a function `label(montagne: list) -> string` that returns the string containing all the skii tracks in the list `montagne`. The returned string must contain for each position, one of the following characters:

- `X`: no track;
- `V`: green track;
- `B`: blue track;
- `R`: red track;
- `N`: black track.

Note:

The returned list has the size `len(montagne) -1`, you do not label the last position.

Example

```
>>> label([0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0])
"XXXVBBBXBNVBBBXNVBXBNRNBXX"
```

Correction:

```
def label(montagne):
        labels = ""

        s = slope(montagne)

        for i in range(len(montagne) - 1):
                if montagne[i] > 3 and montagne[i] < 12:
                        if abs(s[i]) == 1:
                                labels = labels + "V"
                        elif abs(s[i]) == 2:
                                labels = labels + "B"
                        elif abs(s[i]) == 3:
                                labels = labels + "R"
                        elif abs(s[i]) > 3:
                                labels = labels + "N"
                        else:
                                labels = labels + "X"
                else:
                    labels = labels + "X"
        return labels
```

3.8 Relocation - Principle (0.5 point)

Thanks to global warming, there is no more snow in your resort! You decide to move your skii resort to China. As everyone knows, china is on the opposite site of the globe, the heights there are inverted: the summits become valleys, ascents become descents, plains become plateau, the highest point becomes the lowest point and *vice versa*.

Give a method that translates your mountain in China.

Correction:
You must find the highest value of the list, it will become the new 0. Then for each position, do higest height - current height.

3.9 Relocation - Code (0.5 point)

Write a function `relocate(montagne: list) -> list` that returns the shape of your mountain in china.

Example

```
>>> relocate([0,1,2,6,7,9,11,13,11,9,4,5,7,5,3,6,10,11,13,9,11,7,10,5,3,1,0])
[13,12,11,7,6,4,2,0,2,4,9,8,6,8,10,7,3,2,0,4,2,6,3,8,10,12,13]
```

Correction:

```
def relocate(montagne):
        reloc = []

        m = max(montagne)
        for e in montagne:
                reloc = reloc + [m - e]
        return reloc
```