Méthodes de conception de programmes
Devoir 1

Groupe J

Auteurs :
Cezary Swierzewski
Jérôme Lechat

NOMA :
1114-18-00
5035-18-00

1 Introduction

L'objectif de ce premier devoir dans le cadre du cours de Méthodes de Conception de Programmes consiste à mettre au point un algorithme nommé **find_repeat(a, n)** prenant comme paramètres :

a : une liste de caractères quelconques

n : le nombre d'éléments identiques recherchés dans a

L'algorithme **find_repeat(a, n)** retourne donc la position de la première occurrence d'une répétition de n caracteres dans a. Par exemple, **find_repeat("merciii xxx", 3)** retourne **4** et **find_repeat("merciii xxx", 4)** retourne **-1**.

2 Implémentation de notre solution

Pour ce devoir, nous avons choisi une implémentation en Java... voici notre algorithme :

```
public static int find_repeat(String a, int n) {

    /*
     * @Pre : prend en compte a (une chaine de caracteres
        quelconques) et n (le nombre d'elements identiques
        dans a)
     * @Post : return la position de la premiere occurence
        d'une repetition de n elements identiques dans a
    */


    int len_list = a.length();
    int counter = 1;
    int index = 1;
    char sample = a.charAt(0);


    /* Boucle 1 */
    while(index < len_list){
        if(a.charAt(index) == sample){
            counter++;
        } else {
            sample = a.charAt(index);
            counter = 1;
        }
        if(counter == n){
            return index - counter +1;
        }

        index++;

    }

    return -1;

}
```

3 Spécification complète et formelle

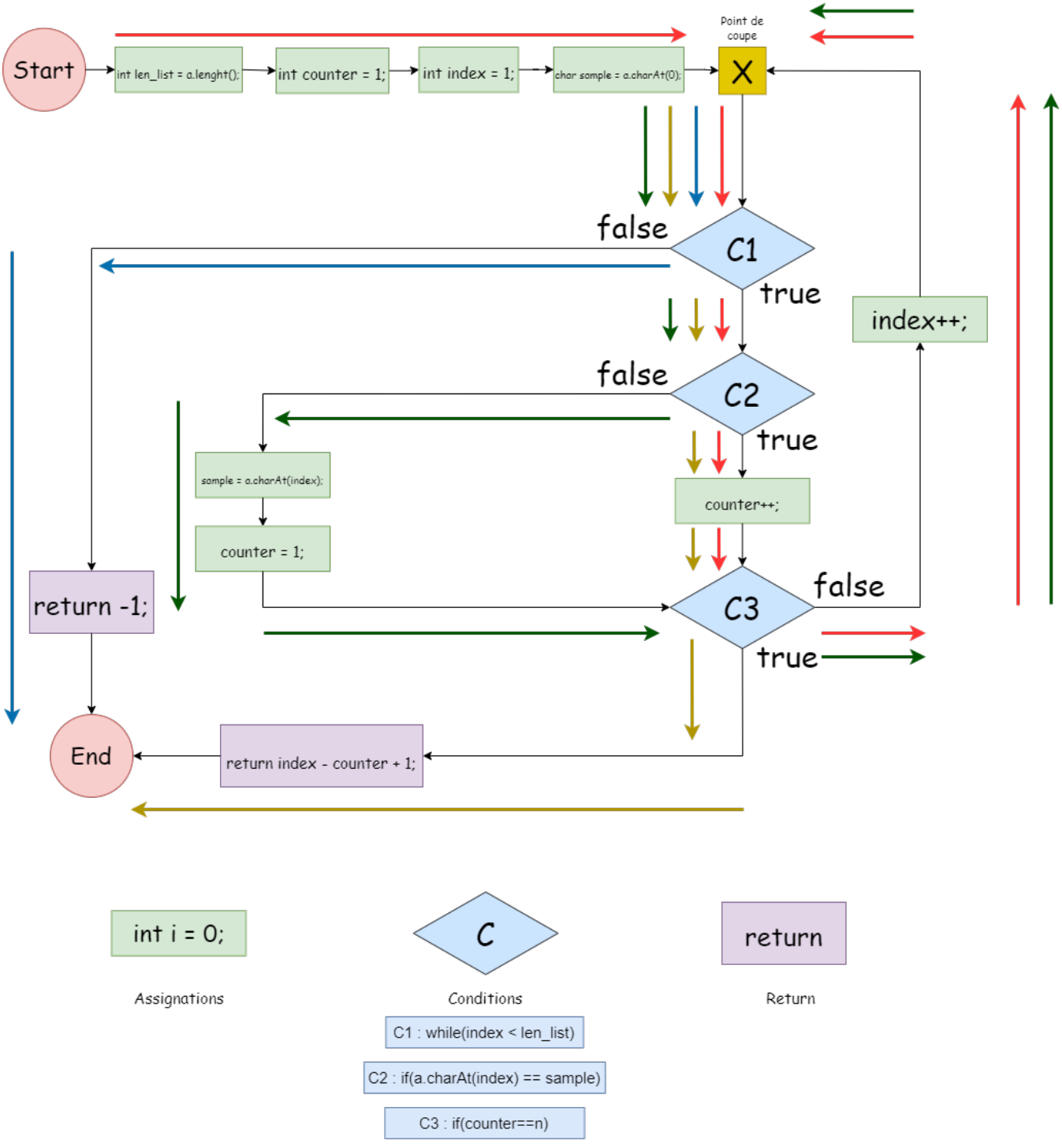
@Pre : $0 < n \leqslant |a| \ \wedge \ |a| > 0 \Longrightarrow 0 < n \leqslant |a|$

@Post : $(Result = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (Result = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)$

Invariant : $1 \leqslant index < |a| \ \wedge \ counter \leqslant n$

Variant : $|a| - index$

4 Graphe du programme avec les points de coupe



5 Preuve

Assertions inductives

Chemin 1

$@Pre[0 < n \leqslant |a| \ \wedge \ |a| > 0 \Longrightarrow 0 < n \leqslant |a|]$
len_list = a.length() ;
$[1 \leqslant 1 < |a| \ \wedge 1 \leqslant n]$
counter = 1 ;
$[1 \leqslant 1 < |a| \ \wedge counter \leqslant n]$
index = 1 ;
sample = a.charAt(0) ;
$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$

Chemin 2

$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$
$[index \geqslant len\_list \Longrightarrow ((result = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (result = -1 \ \wedge \ index == \ |a| \ \wedge \ counter < n))]$
assume index $\geqslant$ len_list ;
$[(-1 = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (-1 = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$
result := -1 ;
$@Post[(Result = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (Result = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$

Chemin 3

$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$
$[index < len\_list \Longrightarrow a.charAt(index) = sample \Longrightarrow counter + 1 == n \Longrightarrow ((index - counter + 1 = index - counter + 1 \wedge counter == n) \vee (index - counter + 1 = -1 \wedge index == |a| \ \wedge \ counter < n))]$
assume index < len_list ;
assume a.charAt(index) = sample ;
$[counter + 1 == n \Longrightarrow ((index - counter + 2 = index - counter + 2 \ \wedge \ counter + 1 == n) \ \vee \ (index - counter + 2 = -1 \ \wedge \ index == |a| \ \wedge \ counter + 1 < n))]$
counter++ ;
$[counter == n \Longrightarrow ((index - counter + 1 = index - counter + 1 \wedge counter == n) \vee (index - counter + 1 = -1 \ \wedge \ index == |a| \ \wedge \ counter < n))]$
assume counter == n ;
$[(index - counter + 1 = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (index - counter + 1 = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$
result := index - counter + 1 ;
$@Post[(Result = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (Result = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$

Chemin 4

$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$
$[index < len\_list \Longrightarrow a.charAt(index) \neq sample \Longrightarrow 1 = n \Longrightarrow ((index - 1 + 1 = index - 1 + 1 \ \wedge \ 1 == n) \ \vee \ (index - 1 + 1 = -1 \ \wedge \ index == \ |a| \ \wedge \ 1 < n))]$
assume index < len_list ;
assume a.charAt(index) $\neq$ sample ;
sample = a.charAt(index) ;
$[1 = n \Longrightarrow ((index - 1 + 1 = index - 1 + 1 \ \wedge \ 1 == n) \ \vee \ (index - 1 + 1 = -1 \ \wedge \ index == |a| \ \wedge \ 1 < n))]$
counter = 1 ;
$[counter = n \Longrightarrow ((index - counter + 1 = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (index -$

$counter + 1 = -1 \ \wedge \ index == |a| \ \wedge \ counter < n))]$
assume counter = n ;
$[(index - counter + 1 = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (index - counter + 1 = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$
result := index - counter + 1 ;
$@Post[(Result = index - counter + 1 \ \wedge \ counter == n) \ \vee \ (Result = -1 \ \wedge \ index == |a| \ \wedge \ counter < n)]$

Chemin 5

$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$
$[index < len\_list \Longrightarrow a.charAt(index) \neq sample \Longrightarrow 1 \neq n \Longrightarrow (1 \leqslant index+1 < |a| \wedge 1 \leqslant n)]$
assume index < len_list ;
assume a.charAt(index) ≠ sample ;
sample = a.charAt(index) ;
$[1 \neq n \Longrightarrow (1 \leqslant index + 1 < |a| \ \wedge \ 1 \leqslant n)]$
counter = 1 ;
$[counter \neq n \Longrightarrow (1 \leqslant index + 1 < |a| \ \wedge \ counter \leqslant n)]$
assume counter ≠ n ;
$[1 \leqslant index + 1 < |a| \ \wedge \ counter \leqslant n]$
index++ ;
$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$

Chemin 6

$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$
$[index < len\_list \Longrightarrow a.charAt(index) = sample \Longrightarrow counter + 1 \neq n \Longrightarrow (1 \leqslant index + 1 < |a| \ \wedge \ counter + 1 \leqslant n)]$
assume index < len_list ;
assume a.charAt(index) = sample ;
$[counter + 1 \neq n \Longrightarrow (1 \leqslant index + 1 < |a| \ \wedge \ counter + 1 \leqslant n)]$
counter++ ;
$[counter \neq n \Longrightarrow (1 \leqslant index + 1 < |a| \ \wedge \ counter \leqslant n)]$
assume counter ≠ n ;
$[1 \leqslant index + 1 < |a| \ \wedge \ counter \leqslant n]$
index++ ;
$@I[1 \leqslant index < |a| \ \wedge \ counter \leqslant n]$

Diminution des variants

$[1 \leqslant index < |a| \ \wedge \ counter \leqslant n \ \wedge \ |a| - index = v_0]$
assume index < len_list ;
assume a.charAt(index) ≠ sample ;
sample = a.charAt(index) ;
counter = 1 ;
assume counter ≠ n ;
index++ ;
$[|a| - index > 0]$

$[1 \leqslant index < |a| \ \wedge \ counter \leqslant n \ \wedge \ |a| - index = v_0]$
assume index < len_list ;
assume a.charAt(index) = sample ;
counter++ ;

assume counter $\neq$ n ;
index++ ;
$[|a| - index > 0]$

6 Conclusion

Ceci conclu donc ce premier devoir dans le cadre du cours de Méthodes de conception de programmes.