**Algorithmique**
J.M. Dischler
Professeur d'informatique

# Master Biologie Structurale et Bioinformatique

TP noté – 14h - 15h, mercredi 12 Octobre 2016

Implémenter les algorithmes à l'aide de Java. En fin de séance, envoyer le fichier contenant les codes sources par e-mail à l'adresse: dischler@unistra.fr.

## Recherche d'une chaine dans une autre chaine

Dans cet exercice nous travaillons avec des chaines de caractères "String". Il existe un certain nombre d'opérations prédéfinies sur les chaines de caractères dans Java, comme la recherche d'une sous-chaine, la concaténation, etc. Cependant nous n'utiliserons pas directement le type String. Nous utiliserons un tableau de caractères. Par exemple:

```
char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
```

définit une chaine "hello". Le point servira à marquer la fin de la chaine.

1.1 Calculer la longueur d'une chaine.

On se propose de "ré-écrire" la fonction : *int length()*.
Ecrire une version itérative de la fonction, puis une **version récursive**. On appellera cette fonction *longueur*. Son profil est: *int longueur(char [] s, int start);*
Par exemple le code suivant:

```
System.out.println( longueur(helloString,0) );
```

affichera le nombre 5 car la chaine "hello" est composée de 5 caractères à partir du caractère d'indice 0. Le "." ne compte pas. Il marque simplement la fin de la chaine.
**Attention**: ne pas trivialement transformer le tableau de caractères en *String* puis appeler length... On utilisera l'opérateur "=" pour comparer des caractères.

1.2 Recherche de la première occurrence d'une sous-chaine.

On se propose de "ré-écrire" la fonction : *int indexOf(String str).*
On proposera une **formulation récursive uniquement**. On appellera cette fonction *rechOcc*. Son profil est : *int rechOcc(char [] s, char [] r, int x);*
Par exemple le code suivant:

```
char [] x = { 'l', 'o', '.' };
System.out.println( rechOcc(helloString, x, 0) );
```

affichera le nombre 4 car "lo" se trouve à partir du 4ème caractère dans la chaine "hello" prise à partir de son premier caractère d'indice 0. Si la sous-chaine n'est pas trouvée, alors la fonction renvoie 0. Par contre la fonction renvoie 2 si l'appel est:

```
System.out.println( rechOcc(helloString, x, 2) );
```

1.3 Recherche du nombre d'occurrences d'une sous-chaine.

En utilisant la fonction précédente, écrire une fonction permettant de calculer le nombre d'occurrences d'une sous-chaine. On appellera cette fonction *nOcc*. Son profil est : *int nOcc(char [] s, char [] r, int x);*
Par exemple le code suivant:

```
char [] y = { 'l', 'o', 'l', 'o', '.' };
System.out.println( rechOcc(y, x, 0) );
```

affichera le nombre 2 car "lo" se trouve deux fois dans y à partir du premier caractère d'indice 0. Par contre l'appel : System.out.println( rechOcc(y, x, 1) ); affichera le nombre 1.

1.4 Evaluer la complexité des fonctions *rechOcc* et *nOcc* en fonction du nombre de caractères des chaines. On comptera notamment le nombre de comparaisons de caractères.