

Épreuve de contrôle continu
du Jeudi 8 octobre 2020

Durée : 1 heure 15

Tous documents autorisés. Il est interdit d'accéder à internet

1 Tri à bulles (4 points)

Voici l'algorithme du tri à bulles

```
tri_à_bulles(T[], n) {
  pour (i de n à 2) {
    pour (j de 1 à i-1) {
      si (T[j + 1] < T[j]) {
        x ← T[j]
        T[j] ← T[j + 1]
        T[j + 1] ← x
      }
    }
  }
}
```

- Exécutez cet algorithme et donnez le nombre de modifications effectuées pour le tableau suivant :

		1	2	3	4	5
		10	2	3	9	8

		1	2	3	4	5
i	j	10	2	3	9	8
5	1	2	10	3	9	8
5	2	2	3	10	9	8
5	3	2	3	9	10	8
5	4	2	3	9	8	10
4	1	2	3	9	8	10
4	2	2	3	9	8	10
4	3	2	3	8	9	10
3	1	2	3	8	9	10
3	2	2	3	8	9	10
2	1	2	3	8	9	10

10 modifications effectuées

.....

2. Quelle est la complexité dans le pire des cas du tri à bulles ?

$O(n^2)$

.....

3. Quelle est la complexité dans le meilleur des cas du tri à bulles ?

$O(n^2)$

.....

2 Liste chaînée (5 points)

Dans le cours, nous avons vu comment parcourir une liste simplement chaînée et y ajouter des éléments. Écrivez l'algorithme `insèreAvant(elt, p, L)` qui insère l'élément `elt` avant l'élément `p` dans la liste `L`. On suppose que `elt` n'est pas déjà dans la liste et que `p` y est.

```
insèreAvant(elt, p, L) {
  si (premier(L) = p) {
    premier(L) <- elt
  } sinon {
    e <- premier(L)
    tant que (suivant(e) ≠ p) {
      e <- suivant(e)
    }
    suivant(e) <- elt
  }
  suivant(elt) <- p
}
```

.....

3 Tableaux circulaires (5 points)

1. Écrivez un algorithme effectuant un décalage circulaire des valeurs d'un tableau d'un cran vers la gauche.

Exemple :

A	B	C	D	E
---	---	---	---	---

→

B	C	D	E	A
---	---	---	---	---

```
décalageGauche(T, n) {
  aux <- T[1]
  pour (i de 2 à n) {
    T[i - 1] <- T[i]
  }
  T[n] <- aux
}
```

.....

2. Écrivez un algorithme effectuant un décalage circulaire des valeurs d'un tableau d'un cran vers la droite.

Exemple :

A	B	C	D	E
---	---	---	---	---

→

E	A	B	C	D
---	---	---	---	---

```
décalageDroite(T, n) {
  aux ← T[n]
  pour (i de n à 2) {
    T[i] ← T[i-1]
  }
  T[1] ← aux
}
```

.....

4 Cryptage d'une chaîne de caractères (6 points)

Pour échanger des messages secrets, vous avez recours à un cryptage afin que le message ne soit pas lisible par une autre personne que votre correspondant.

Une technique de cryptographie consiste à remplacer les caractères selon une clé de substitution. Pour cela, on utilise un alphabet-clé, représenté par un tableau, dans lequel les lettres se succèdent de manière désordonnée, par exemple :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
H	Y	L	U	J	P	V	R	E	A	K	B	N	D	O	F	S	Q	Z	C	W	M	G	I	T	X
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les A deviendront des H, les B des Y, les C des L, ... "BONJOUR" devient "YODAOWQ".

Pour répondre aux questions, vous pouvez utiliser les fonctions suivantes :

- entier positionAlpha(caractère c) qui donne la position d'un caractère dans l'alphabet. Exemple : positionAlpha('I') rend 9.
- caractère caracAlpha(entier e) qui donne le caractère à la position de l'entier dans l'alphabet. Exemple : caracAlpha(13) rend 'M'.

1. Encodez la phrase "LE CODE C'EST LE CODE".

BJ LOUJ L'JZC BJ LOUJ

.....

2. Écrivez la fonction encode prenant en paramètre un alphabet-clé de codage et tableau de caractères, et rendant un tableau de caractères crypté.

```
encode(alphabet, chaine, n) {
  // résultat est un tableau de n cases
  pour (i de 1 à n) {
    résultat[i] ← alphabet[positionAlpha(chaine[i])]
  }
  retourner résultat
}
```

.....

3. Écrivez la fonction `alphaDécodage` prenant en paramètre un alphabet-clé de codage, et rendant l'alphabet-clé de décodage.

Exemple : L'alphabet-clé de décodage correspondant à l'alphabet-clé de codage précédent :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
J	L	T	N	I	P	W	A	X	E	K	C	V	M	O	F	R	H	Q	Y	D	G	U	Z	B	S
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

```
alphaDécodage(alphabet) {
  // résultat est un tableau de 26 cases
  pour (i de 1 à 26) {
    résultat[positionAlpha(alphabet[i])] <- caracAlpha(i)
  }
  retourner résultat
}
```

.....