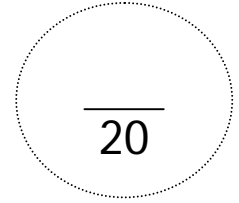


Lycée El Krib	<i>Devoir de contrôle</i> <i>N° 3</i>	Année scolaire : 2008 / 2009	
Professeur : Mohamed TRABELSI		Durée : 1h	Théorique
Matière : Informatique		Classe : 3 <sup>ème</sup> SI	

Nom & prénom : .....



Énoncé : Algorithme de Tri

On désire créer un programme intitulé `tri_vecteur` qui permet de trier un tableau `T` de `n` entiers distincts ( $5 \leq n \leq 20$ ) dans l'ordre décroissant selon le principe suivant :

Pour chaque élément de `T` :

- Déterminer le nombre d'éléments qui lui sont supérieurs.
- Placer cet élément dans un tableau `R` à la case d'indice nombre d'élément plus grand + 1.

Exemple :

<b>T</b>	6	2	0	5	12	25
	1	2	3	4	5	6

Deux valeurs sont supérieures au premier élément de `T`. Cet élément sera donc placé à la position 3 du tableau `R`.

<b>R</b>			6			
	1	2	3	4	5	6

Le résultat affiché est `R` qui sera un tableau trié dans l'ordre décroissant.

### 1. Analyse principale

Résultat : Affichage

Traitement :

Pour `i` de 1 à `n` faire

Ecrire (`R[i]`)

Fin pour

Tri (`T`, `R`, `n`)

Remplir (`T`, `n`)

Saisie (`n`)

### Algorithme

- 0) Début `tri_vecteur`
- 1) Saisie (`n`)
- 2) Remplir (`T`, `n`)
- 3) Tri (`T`, `R`, `n`)
- 4) Pour `i` de 1 à `n` faire
  - Ecrire (`R[i]`)
  - Fin pour
- 5) Fin `tri_vecteur`

### T.A.F

1. Faire les déclarations nécessaires. (4 pts)
2. Faire l'analyse et l'algorithme des sous programmes suivants :
  - Saisie ( ) (3 pts)
  - Remplir ( ) (4 pts)
  - Tri ( ) (9 pts)

Remarque :

D'autres sous programmes pourraient être envisagés en cas de besoin.

Lycée El Krib	<i>Correction</i> <i>Devoir de contrôle</i> <i>N° 3</i>	Année scolaire : 2008 / 2009	
Professeur : Mohamed TRABELSI		Durée : 1h	Théorique
Matière : Informatique		Classe : 3 <sup>ème</sup> SI	

### T.D.O Globaux

Type
Tab = tableau de 20 entiers

Objet	Type	Rôle
T	Tab	
R	Tab	
n	octet	
i	octet	compteur

### 2. Analyse de la procédure Saisie

Procédure saisie (var n : octet)

Résultat : n

Traitement :

Répéter

n = donnée (" Donner n entre 5 et 20 : ")

Jusqu'à n dans [5..20]

### Algorithme

0) Procédure saisie (var n : octet)

1) Répéter

Écrire (" Donner n entre 5 et 20 : "), Lire (n)

Jusqu'à n dans [5..20]

2) Fin

### 3. Analyse de la procédure Remplir

Procédure remplir (var T : Tab ; n : octet)

Résultat : T

Traitement :

Pour i de 1 à n faire

T[i] = donnée (" Donner la case ", i, " : ")

Fin Pour

### Algorithme

0) Procédure remplir (var T : Tab ; n : octet)

1) Pour i de 1 à n faire

Ecrire (" Donner la case ", i, " : "), Lire (T[i])

Fin Pour

2) Fin

### T.D.O Locaux

Objet	Type	Rôle
i	Octet	compteur

#### 4. Analyse de la procédure Tri

Procédure tri (T : Tab ; var R : Tab ; n : octet)

Résultat : R

Traitement :

```
Pour i de 1 à n faire
    X ← nbr_grand (T, n, i) + 1
    R[x] ← T[i]
Fin pour
```

#### Algorithme

0) Procédure tri (T : Tab ; var R : Tab ; n : octet)

1) Pour i de 1 à n faire

```
X ← nbr_grand (T, n, i) + 1
```

```
R[x] ← T[i]
```

Fin pour

2) Fin

T.D.O Locaux

Objet	Type	Rôle
i	Octet	compteur
x	Octet	Position d'insertion dans R

#### 5. Analyse de la fonction nbr\_grand

Fonction nbr\_grand (T : Tab ; n : octet ; i : octet) : octet

Résultat : nbr\_grand

Traitement :

```
nbr_grand ← cpt
cpt ← 0
Pour j de 1 à n faire
    Si (j ≠ i) et T[j]>T[i] Alors cpt ← cpt + 1
    Fin si
Fin pour
```

## Algorithme

0) Fonction **nbr\_grand** (T : Tab ; n : octet ; i : octet) : octet

1)  $cpt \leftarrow 0$

Pour j de 1 à n faire

**Si** (j ≠ i) et T[j]>T[i] **Alors**  $cpt \leftarrow cpt + 1$

**Fin si**

Fin pour

2)  $nbr\_grand \leftarrow cpt$

3) Fin

T.D.O Locaux

Objet	Type	Rôle
j	Octet	compteur
cpt	Octet	Nombre d'éléments plus grands que T [i]