PROTOTYPE
2022

Épreuve d'informatique

Sections : **Mathématiques, Sciences expérimentales et Sciences techniques**

Durée : **1 heure 30 mn** | Coefficient : **0.5**

Exercice 1 *(2 points)*

On veut déterminer et afficher le nombre de diviseurs d'un entier **n** strictement positif.

Exemple :

Pour **n=6**, le nombre de diviseurs de **6** est égal à **4**. En effet les diviseurs de **6** sont {**1** ; **2** ; **3** ; **6**}.

On vous propose les trois séquences d'instructions algorithmiques suivantes :

Séquence 1	*Séquence 2*	*Séquence 3*

c ← 0
Pour k de 1 à n faire
  Si (n mod k = 0) alors
  c ← c + 1
  Fin Si
Fin Pour
Ecrire (c)
	c ← 1
	Pour k de 2 à (n div 2) faire
	  Si (n mod k = 0) alors
	  c ← c + 1
	  Fin Si
	Fin Pour
	Ecrire (c)
		c ← 2
		Pour k de 2 à (n-1) faire
		  Si (n mod k = 0) alors
		  c ← c + 1
		  Fin Si
		Fin pour
		Ecrire (c)

1) Compléter le tableau ci-dessous par la valeur de la variable **c** après exécution de chaque séquence, et ce pour **n=4**.

Séquence	Valeur de la variable c
1	
2	
3	

2) Donner le numéro de la séquence qui ne permet pas d'afficher le nombre de diviseurs. Justifier votre réponse.

.....

.....

Exercice 2 *(3 points)*

Un médecin veut chercher la fiche d'un de ses patients en connaissant son nom. Pour cela, il utilise un tableau **T** contenant **N** noms.

1) Compléter la séquence algorithmique présentée ci-dessous afin de vérifier l'existence d'un nom donné **NOM** dans un tableau **T** non vide.

```
Algorithme recherche
Début
  Écrire ("Donner le nom à chercher : ")
  Lire ( .....)
  Existe ← ……..
  i ← …….
  Répéter
    Si (T[i] = NOM) alors
          Existe ← ……
    Sinon
          i← ……..
    Finsi
  Jusqu'à (……) ou (……..)
  Si (……..) alors
        Écrire ("Le nom recherché existe dans ce tableau.")
  Sinon
        Écrire ("Le nom recherché n'existe pas dans ce tableau.")
  Finsi
Fin
```

T.D.O.	
Objet	**Type/Nature**
T	Tableau de N chaînes
N, i	Entier
NOM	Chaîne
Existe	Booléen

2) Ce médecin veut chercher les numéros des fiches de ses patients ayant le même nom. Modifier la séquence algorithmique présentée ci-dessus afin d'afficher ces numéros.

.....

.....

.....

.....

.....

.....

.....

.....

Exercice 3 *(5 points)*

La propagation de l'épidémie Covid-19 suit une croissance exponentielle. Pour déterminer et afficher le nombre total de personnes contaminées pendant un nombre de jours donné **N** et pour **x** personnes initialement contaminées on utilise la formule suivante :

$$e^x = \sum_{i=0}^{N} \frac{(x)^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots \ldots \ldots \ldots + \frac{(x)^N}{N!}$$

Donner un algorithme solution à ce problème.

Exercice 4 *(10 points)*

Un débutant en anglais veut élaborer son propre **carnet** comme étant un dictionnaire **FRANÇAIS/ANGLAIS** pour l'utiliser afin de traduire des phrases au cours de sa formation. Pour cela, on admettra qu'un traducteur du français à l'anglais peut être simplifié par une traduction de mot-à-mot.

Ce carnet **FRANÇAIS/ANGLAIS** est formé par **N** mots en français et par **N** mots en anglais, de sorte que chaque mot en français lui correspond son équivalent en anglais, avec $\mathbf{2 \leq N \leq NMAX}$ (**NMAX** est une constante égale à **100**).

Pour chaque mot ajouté en français, dans le carnet, en lui ajoute en même temps son équivalent en anglais. Sachant que chaque mot en français qu'en anglais est une chaîne non vide de longueur maximale **15** lettres non accentuées.

Après l'élaboration du carnet, on veut traduire une phrase donnée en français (formée seulement par des lettres et par des espaces) en son équivalent en anglais. Dans le cas où l'un des mots ne figure pas dans le carnet, le mot en français va figurer dans la phrase en anglais mais entre deux accolades.

La phrase traduite doit être suivie par un message de succès ou un message d'échec indiquant le nombre de mots non traduits.

NB : On suppose que la phrase ne contient pas un espace au début, un espace à la fin et des espaces superflus (un seul espace sépare deux mots).

Exemple : Pour **N=5**, le carnet sera présenté comme suit :

FR	ecole	je	libre	suis	un
	0	1	2	3	4
ENG	School	i	free	am	a
	0	1	2	3	4

FR contient les mots en français et **ENG** contient les mots en anglais.

- Pour la phrase **"ecole libre"**
  Le résultat affiché sera : **"school free : Traduction totale "**
- Pour la phrase **"je suis un etre libre"**
  Le résultat affiché sera : **"i am a {etre} free : Traduction partielle, 1 mot(s) non traduit(s)"**

On vous demande d'élaborer :

1) un algorithme du programme principal, solution à ce problème, en le décomposant en modules,
2) l'algorithme de chaque module.

PROTOTYPE
CORREDTION
2022

Épreuve d'informatique	
Sections : **Mathématiques, Sciences expérimentales et Sciences techniques**	
Durée : **1 heure 30 mn**	Coefficient : **0.5**

CORRECTION

EXERCICE 1

1) Compléter le tableau ci-dessous par la valeur de la variable **c** après exécution de chaque séquence, et ce pour **n=4**.

Séquence	Valeur de la variable c
1	3
2	2
3	3

2) Donner le numéro de la séquence qui ne permet pas d'afficher le nombre de diviseurs. Justifier votre réponse.

la séquence 2 ne permet pas d'afficher le nombre de diviseurs:
k commence de 2 a n div 2 ==> On a eliminé les 2 diviseurs 1 et n
alors, il faut initialiser c par 2 (c<-- 2)

EXERCICE 2

1)

```
Algorithme recherche
Début
  Écrire ("Donner le nom à chercher : ")
  Lire ( NOM )
  Existe ← Faux
  i ← 0
  Répéter
    Si (T[i] = NOM) alors
          Existe ← Vrai
    Sinon
          i← i + 1
    Finsi
  Jusqu'à ( Existe ) ou ( i=N )
  Si ( Existe ) alors
        Écrire ("Le nom recherché existe dans ce tableau.")
  Sinon
        Écrire ("Le nom recherché n'existe pas dans ce tableau.")
  Finsi
Fin
```

2)

```
Algorithme recherche
Début
    Écrire ("Donner le nom à chercher : ")
    Lire ( NOM )
    Existe ←Faux
    Pour i de 0 a N-1 faire
      Si (T[i] = NOM) alors
            Ecrire("le numéro de ",NOM," est ",i )
            Existe ← Vrai
      Finsi
    Fin pour
    Si ( Existe = Faux ) alors
      Écrire ("Le nom recherché n'existe pas dans ce tableau.")
    Finsi
Fin
```

EXERCICE 3

Algorithme de la fonction somme

```
Fonction somme(N :entier, x : entier) :réel
Debut
        S←0
        Pour i de 0 a N faire
                S←S+ Pow(x,i)/Fact(i)
        Fin pour
        Retourner S
Fin
```

TDOL

Objet	Type/Natur
S	Reel
i	Entier
Pow,Fact	fonction

Algorithme de la fonction Pow

```
Fonction Pow(x :entier, y : entier) :réel
Debut
        P←1
        Pour i de 0 a y faire
                P←P*x
        Fin pour
        Retourner P
Fin
```

TDOL

Objet	Type/Natur
P	Entier
i	Entier

Algorithme de la fonction Fact

```
Fonction Fact(n : entier) :entier
Debut
        F←1
        Si N=0 alors
                Retourner 1
        Si non
                Pour i de 1 a N+1 faire
                        F←F*i
                Fin pour
                Retourner F
        Fin si
Fin
```

TDOL

Objet	Type/Natur
F	Entier
i	Entier

EXERCICE 4 : CORRECTION PROBLÈME

Algorithme du programme principal

Début Problème

N←**saisie**()

remplirFR_ENG(FR,ENG,N)

ch←**saisiech**()

chtr←**Traduire**(FR,ENG,ch,N)

Ecrire(chtr)

Fin

TDNT

Type
Tab=tableau de NMAX chaine de caractères

TDOG

Objet	Type/Natur
N	Entier
ch , chtr	Chaine
NMAX	Constante=100
FR,ENG	Tab
Saisie saisiech Traduire	Fonction
remplirFR_ENG	Procédures

Algorithme de la fonction saisie

Fonction saisie() : entier

Debut

Repeter

Ecrir("donner N ")

Lire(N)

Jusqu'à 2≤N≤NMAX

Retourner N

Fin

TDOL

Objet	Type/Natur
N	Entier

Algorithme de la procedure remplirFR_ENG

Procedure **remplirFR_ENG**(@FR:Tab,@ENG:Tab,N:entier)

Debut

Pour i de 0 a N-1 faire

Repeter

Ecrire(" FR[" , i , "]= ")

Lire(FR[i])

Jusqu'a 1<=long(FR[i])<=15 et Verif(FR[i])

Repeater

Ecrire(" ENG[" , i , "]= ")

Lire(ENG[i])

Jusqu'a 1<=long(ENG[i])<=15 et Verif(ENG[i])

Fin pour

Fin

TDOL

Objet	Type/Natur
.i	Entier
Verif	Fonction

Algorithme de la fonction Verif

```
fonction Verif(ch :chaine ) :booléen
Debut
Cond←vrai
Pour i de 0 a long(ch)-1 faire
        Si   NON("a"<=ch[i]<= "z" ou "A"<=ch[i]<= "Z" )
                Cond←faux
        Fin si
Fin pour
Retourner cond
Fin
```

TDOL

Objet	Type/Natur
i	Entier
Cond	Booléen

Algorithme de la fonction Traduire

```
fonction Traduire(FR,ENG:Tab,ch:chaine,N:entier):chaine
Debut
Trad←""
K←0
Tant que ch ≠"" faire
        P←pos(" ",ch)
        Si P=-1 alors
                Mot←ch
                Ch←""
        Sinon
                Mot←sous_chaine(ch,0,pos)
        Fin si
        Test←faux
        Pour i de 0 a N faire
                Si Mot=FR[i] alors
                        Trad=trad+" "+ENG[i]
                        Test←vrai
                Fin si
        Fin pour
        Si Test=faux alors
                Trad=trad+" "+"{"+Mot+"}"
                K←K+1
        Fin si
        Ch←sous_chaine(ch,P+1,long(ch))
Fin tant que
Trad←sous_chaine(trad,1,long(Trad))
 Si  k=0 alors
     Retourner  " « "+trad+" : Traduction totale »"
 Sinon
     Retourner  " « "+trad+ " : Traduction partielle, "+convch(k)+ " mot(s) non traduit(s) »"
Fin si
Fin
```

TDOL

Objet	Type/Natur
Trad , Mot	chaines
P , k	Entiers
Test	Booleen