| RÉPUBLIQUE TUNISIENNE<br>MINISTÈRE DE L'ÉDUCATION<br>●●○●●<br>EXAMEN DU BACCALAURÉAT<br>**SESSION 2017** | *Épreuve* : **Algorithmique et Programmation** | |
|---|---|---|
| | *Section* : **Sciences de l'informatique** | |
| | *Durée* : **3h** | Coefficient : **2.25** |
| | ***Session principale*** | |

## Exercice 1 : (2,25 Points)

Soit l'algorithme ci-dessous de la fonction **Rectangle** permettant de calculer, en utilisant la méthode des rectangles, l'aire résultante de la courbe de la fonction $f(x) = \frac{6}{1+x}$ sur un intervalle **[a,b]** subdivisé en **n** rectangles.

```
0) DEF FN Rectangle (a, b : Réel ; n : Entier) : Réel
1) h ← (b – a)/n
   S ← 0
   x ← a
   Pour i de 1 à n Faire
          S ← S + 6/(1+x)
          x ← x + h
   Fin Pour
2) Rectangle ← S * h
3) FIN Rectangle
```

**Travail demandé :**

Pour chacune des questions suivantes, valider chaque proposition par **V** si la réponse est correcte ou par **F** dans le cas contraire.

**1)** La fonction **Rectangle** permet de calculer l'aire résultante de la courbe de la fonction **f** sur un intervalle [a,b] selon la méthode des :

| | |
|---|---|
| ☐ | **rectangles à gauche** |
| ☐ | **rectangles du point milieu** |
| ☐ | **rectangles à droite** |

**2)** Pour les valeurs **a = 1**, **b = 5** et **n = 4**, le résultat retourné par la fonction **Rectangle** est :

| | |
|---|---|
| ☐ | **5.5** |
| ☐ | **7.7** |
| ☐ | **10.12** |

**3)** Pour appliquer la méthode des trapèzes au lieu de la méthode des rectangles, on remplace l'instruction de calcul de la somme **S** par :

| | |
|---|---|
| ☐ | **S ← S + (6/(1+$x$) + 6/(1+$x$+h))/2** |
| ☐ | **S ← S + 6/(1+$x$+h)/ 2** |
| ☐ | **S ← S + (6/(1+$x$) - 6/(1+$x$+h))/2** |

## Exercice 2 : (2,75 points)

Soit $x$ un réel de l'intervalle **]0, 1[**.

En binaire, $x$ s'écrit sur **n** chiffres après la virgule comme suit : $\mathbf{0.c_1c_2c_3c_4c_5...c_{n-1}c_n}$ avec $\mathbf{c_i}$ un chiffre binaire (0 ou 1).

Pour déterminer les chiffres $\mathbf{c_i}$ après la virgule de l'équivalent binaire du réel $x$, on suit le procédé suivant :

1. calculer $\mathbf{c_1}$ en multipliant $x$ par 2,
   - si $\mathbf{2 * x < 1}$, alors $\mathbf{c_1}$ est égal à **zéro** et on remplace $x$ par $\mathbf{2 * x}$
   - si $\mathbf{2 * x \geq 1}$, alors $\mathbf{c_1}$ est égal à **1** et on remplace $x$ par $\mathbf{2 * x - 1}$
2. répéter **n** fois l'étape **1** jusqu'à calculer $\mathbf{c_n}$.

**Exemple 1 :**

Pour $x$ = **0.825** et **n = 5,** l'équivalent binaire de $x$ est $0.c_1c_2c_3c_4c_5$ et se calcule comme suit :

- 2 * 0.825 = 1.65 d'où $c_1 = 1$ et on remplace $x$ par 0.65 (1.65 – 1)
- 2 * 0.65 = 1.3 d'où $c_2 = 1$ et on remplace $x$ par 0.3 (1.3 – 1)
- 2 * 0.3 = 0.6 d'où $c_3 = 0$ et on remplace $x$ par 0.6 (2*0.3)
- 2 * 0.6 = 1.2 d'où $c_4 = 1$ et on remplace $x$ par 0.2 (1.2 – 1)
- 2 * 0.2 = 0.4 d'où $c_5 = 0$

D'où l'équivalent binaire à **5** chiffres après la virgule de **0.825** est **0.11010**

**Exemple 2 :**

Pour $x$ = **0.625** et **n = 4,** l'équivalent binaire de $x$ est $0.c_1c_2c_3c_4$ et se calcule comme suit :

- 2 * 0.625 = 1.25 d'où $c_1 = 1$ et on remplace $x$ par 0.25 (1.25 -1)
- 2 * 0. 25 = 0.5 d'où $c_2 = 0$ et on remplace $x$ par 0.5 (0.25*2)
- 2 * 0.5 = 1.0 d'où $c_3 = 1$ et on remplace $x$ par 0 (1.0 - 1)
- 2 * 0 = 0 d'où $c_4 = 0$

D'où l'équivalent binaire à **4** chiffres après la virgule de **0.625** est **0.1010**

**Travail demandé :**

Ecrire un algorithme d'une fonction qui retourne la représentation binaire, sur **n** chiffres après la virgule, d'un réel $x$ de l'intervalle **]0, 1[**.

**NB :**

- $x$ et **n** sont passés en paramètres et ils sont déjà saisis dans le module appelant.
- Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

| Objet | Type / Nature | Rôle |
|---|---|---|
| | | |

## Exercice 3 : (5 points)

Pour **A** et **B** deux entiers strictement positifs, on définit la relation suivante :

**Si (A! * B! mod (A+B) = A) OU (A! * B! mod (A+B) = B)**

**alors (A+B) est un nombre premier**

*(Avec A! et B! sont respectivement la factorielle de A et celle de B)*

**Exemples :**

| A | B | A+B | A! | B! | A! * B! | A! * B! mod (A+B) | A+B |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 5 | 2 | 6 | 12 | 12 mod 5 = 2 (= A) | 2+3 = 5 est premier |
| 7 | 4 | 11 | 5040 | 24 | 120960 | 120960 mod 11 = 4 (= B) | 7+4 = 11 est premier |
| 5 | 2 | 7 | 120 | 2 | 240 | 240 mod 7 = 2 (= B) | 5+2 = 7 est premier |

**Travail demandé :**

En disposant d'un fichier texte nommé **"Source.txt"** contenant dans chaque ligne un couple de deux valeurs séparées par un espace représentant respectivement les valeurs de deux entiers **A** et **B**, écrire un algorithme d'un module, qui à partir du fichier **"Source.txt"**, permet :

1. de générer un nouveau fichier texte **"Resultat.txt"** contenant dans chaque ligne, les valeurs du couple **A** et **B**, vérifiant la relation définie précédemment, sous la forme d'un nombre complexe comme suit : **"A+i*B"**
2. d'afficher le contenu du fichier **"Resultat.txt"**

**Exemple :**

Pour le contenu du fichier **"Source.txt"** suivant :

```
2 3
4 5
7 4
6 3
5 2
```

Le fichier **"Resultat.txt"** aura le contenu suivant :

```
2+i*3
7+i*4
5+i*2
```

## Problème : (10 points)

On se propose de réaliser un moteur de recherche local permettant de trouver, sur un ordinateur, tous les fichiers textes contenant un ensemble de mots saisis par l'utilisateur.

Pour cela, on dispose d'un fichier texte nommé **"Chemin.txt"** situé sur la racine du disque C et contenant les chemins d'accès des fichiers textes du disque local de l'ordinateur à raison d'un chemin par ligne, sachant que :

- un chemin d'accès est composé d'au maximum **80** caractères
- le nombre maximum de fichiers texte est égal à **100**

Le procédé de recherche consiste à :

- saisir dans un tableau **TM** les **N** mots (**0 < N < 11**) à rechercher. Un mot est formé uniquement par des lettres,
- remplir une matrice **M** par le nombre de lignes, contenant le mot à rechercher, dans chaque fichier tels que :

  **M[i,j]** = nombre de lignes du fichier **G** contenant le mot à rechercher **TM[j]**
  Où **G** représente le fichier dont son chemin est indiqué dans la ligne numéro **i** du fichier **"Chemin.txt"**
- afficher tous les mots à rechercher suivis par les chemins des fichiers qui les contiennent s'ils existent séparés par un espace.

**Exemple :**

Pour :

- le fichier "**Chemin.txt**" suivant :

| |
|---|
| *C:\bac2017\matieres.txt*<br>*C:\bac2017\programs.txt*<br>*C:\divers\textes\web.txt*<br>*C:\revision.txt*<br>*C:\Application\Exercice.txt* |

- **N = 4** et le tableau **TM** suivant :

| | | | |
|---|---|---|---|
| *Informatique* | *Algorithme* | *Html* | *Php* |

Si la recherche des nombres d'occurrences des mots clés donne la matrice **M** suivante :

| | *1* | *2* | *3* | *4* |
|---|---|---|---|---|
| *1* | 0 | **1** | 0 | 0 |
| *2* | **1** | 0 | 0 | 0 |
| *3* | 0 | 0 | 0 | **4** |
| *4* | 0 | 0 | 0 | 0 |
| *5* | 0 | 0 | 0 | **10** |

Alors le programme Affichera :

**Informatique :** *C:\bac2017\Programs.txt*
**Algorithme :** *C:\bac2017\Matieres.txt*
**Html :**
**Php:** *C:\divers\textes\Web.txt* *C:\Application\Exercice.txt*

**Travail demandé :**

1- Analyser le problème en le décomposant en modules.
2- Ecrire un algorithme solution pour chaque module envisagé. Chaque algorithme proposé doit être accompagné d'un tableau de déclaration des objets ayant la forme suivante :

| Objet | Type / Nature | Rôle |
|---|---|---|
| | | |