

Section : N° d'inscription : Série :

Nom et prénom :

Date et lieu de naissance :

Signature des
surveillants

Épreuve : *Algorithmique et Programmation* - Section : *Sciences de l'informatique* - Session *principale 2020*

Feuille à compléter par le candidat et à rendre avec sa copie.

Important :

Chaque solution développée par le candidat sous forme d'un algorithme doit être accompagnée d'un tableau de déclaration des objets ayant la forme suivante :

Objet	Type / Nature	Rôle

Exercice 1 (2,25 points)

Valider chacune des propositions suivantes par la lettre "V" si elle est correcte ou par la lettre "F" dans le cas contraire.

1) Pour une variable de type enregistrement :


- ☐ Deux champs peuvent avoir le même type
- ☐ Deux champs peuvent avoir le même nom
- ☐ Le nombre de champs peut dépasser 3

2) Pour un fichier de type texte :

- ☐ L'accès est séquentiel
- ☐ Chaque ligne est formée uniquement par des lettres
- ☐ Le stockage est permanent

3) Dans un système de numération de base **B** ($B \geq 2$), le nombre décimal **B+1** est représenté par :

- ☐ 10
- ☐ 11
- ☐ Cela dépend de B

RÉPUBLIQUE TUNISIENNE MINISTÈRE DE L'ÉDUCATION EXAMEN DU BACCALAURÉAT SESSION 2020	Session principale	
	 Épreuve : Algorithmique et Programmation	Section : Sciences de l'informatique
	Durée : 3h	Coefficient de l'épreuve : 2.25

⌘ ⌘ ⌘ ⌘ ⌘ ⌘

*Le sujet comporte 4 pages numérotées de 1/4 à 4/4.
La page 1/4 est à remplir par le candidat et à rendre avec sa copie*

Exercice 2 : (3,5 points)

Soient k, n deux entiers naturels et U une suite définie par :

$$\begin{cases} U_0 = 1 \\ U_{n+1} = k * U_n \end{cases}$$

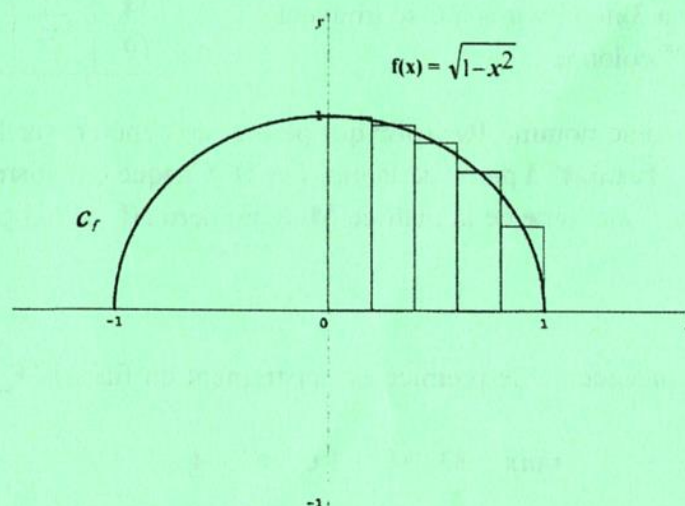
- 1) Quel est l'ordre de récurrence de la suite U ? Justifiez votre réponse.
- 2) Calculer les termes U_1, U_2 et U_3 pour $k = 4$.
- 3) Parmi les trois propositions suivantes, réécrire sur votre feuille de copie celle qui correspond au rôle de la suite U :
 - La suite U permet de calculer la factorielle de k ($k!$)
 - La suite U permet de calculer k à la puissance n (k^n)
 - La suite U permet de calculer le produit de n et k ($n * k$)
- 4) Écrire un algorithme d'une fonction récursive qui permet de calculer le terme U_n pour tout entier n supérieur ou égale à zéro.

N.B. : L'entier n est saisi dans le programme appelant.

Exercice 3 : (5,5 points)

Soit la fonction f définie par $f(x) = \sqrt{1-x^2}$ pour tout $x \in [-1,1]$.

Pour calculer une valeur approchée de $\int_0^1 f(x)dx$ en utilisant la méthode des rectangles, on commence par subdiviser l'intervalle $[0,1]$ en n sous-intervalles, puis on additionne les surfaces des rectangles. La figure ci-après représente la courbe C_f de la fonction f où l'intervalle $[0,1]$ a été subdivisé en cinq sous-intervalles.



Voir suite au verso ➡

1) Parmi les trois méthodes suivantes, réécrire sur votre feuille de copie celle illustrée dans les subdivisions représentées dans la figure précédente et justifiez votre réponse.

- La méthode des rectangles à gauche
- La méthode des rectangles à droite
- La méthode des rectangles du point milieu

2) On se propose de calculer une valeur approchée de π en utilisant les deux formules suivantes :

Formule 1 :

$$\frac{\pi}{4} = \int_0^1 f(x) dx \quad \text{avec} \quad f(x) = \sqrt{1-x^2}$$

Formule 2 :

$$\pi = 16 * \left(\frac{1}{5^1} - \frac{1}{3} * \frac{1}{5^3} + \frac{1}{5} * \frac{1}{5^5} - \frac{1}{7} * \frac{1}{5^7} + \dots \right) - 4 * \left(\frac{1}{239^1} - \frac{1}{3} * \frac{1}{239^3} + \frac{1}{5} * \frac{1}{239^5} - \frac{1}{7} * \frac{1}{239^7} + \dots \right)$$

- a) Ecrire un algorithme d'un module nommé **Valeur1** qui permet de calculer une valeur approchée de π à 10^{-4} près en appliquant la **Formule 1** et en utilisant la méthode de calcul choisie dans la question 1).
- b) Ecrire un algorithme d'un module nommé **Valeur2** qui permet de calculer une valeur approchée de π à 10^{-4} près en utilisant la **Formule 2**.
- c) Ecrire un algorithme d'un module nommé **Compare** qui affiche parmi les deux formules précédentes celle dont la valeur approchée trouvée est la plus proche de la valeur d'une constante $\pi = 3.1415$

Exercice 4 : (5 points)

Soit **M** une matrice carrée de **n x n** entiers distincts avec **n** \in **[5,50]**, chaque ligne contient un seul maximum. On désigne par colonne dominante d'une matrice, la colonne qui contient le plus de maximums des lignes de cette matrice.

Exemple :

Pour **n = 5** et la matrice **M** ci-contre, la 4^{ème} colonne est la colonne dominante puisque parmi les 5 maximums des 5 lignes il y a 3 maximums qui se trouvent dans la 4^{ème} colonne.

	1	2	3	4	5
1	32	12	10	89	15
2	3	33	14	1	18
3	54	5	22	76	50
4	34	21	6	29	17
5	19	9	11	84	25

On se propose d'écrire un module nommé **Remplir** qui permet de générer, sur la racine du disque **D**, un fichier d'enregistrements "**F_Max.dat**" à partir de la matrice **M**. Chaque enregistrement du fichier contient la valeur **Vmax** du maximum d'une ligne de la matrice **M**, le numéro **NL** de la ligne et le numéro **NC** de la colonne de **Vmax**.

Exemple :

Pour la matrice précédente, le premier enregistrement du fichier "**F_Max.dat**" contiendra les valeurs suivantes :

$$V_{\max} = 89, NL = 1 \text{ et } NC = 4$$

Travail demandé :

- 1) Dresser un tableau de déclaration des nouveaux types pour déclarer :
 - Un type pour le fichier "**F_Max.dat**" ainsi que tous les nouveaux types nécessaires à sa déclaration.
 - Un type pour la matrice **M**.
- 2) Donner une instruction d'association du fichier "**F_Max.dat**" à une variable logique **F**.
- 3) Développer le module **Remplir**, sachant que **M** et **n** sont déjà saisis au niveau du programme appelant.
- 4) En utilisant une fonction **Frequence (F, NC)** qui retourne le nombre d'occurrences d'un numéro de colonne **NC** dans le fichier d'enregistrements **F**, écrire un algorithme d'un module qui permet de chercher le numéro de la colonne dominante à partir du fichier "**F_Max.dat**".

N.B. :

- Le candidat n'est pas appelé à développer la fonction **Frequence**.
- Lorsque plus qu'une colonne est dominante, on affichera la dernière rencontrée dans le fichier.

Exercice 5 : (3,75 points)

Le tri par comptage dans l'ordre croissant d'un tableau **T1** d'éléments distincts consiste à déterminer pour chaque élément de **T1** sa position dans un tableau trié **T2** qui est égale au nombre d'éléments de **T1** qui lui sont strictement inférieures, auquel on ajoute 1.

Exemple :

T1 =	15	2	14	7	9	17	11
	1	2	3	4	5	6	7
T2 =	2	7	9	11	14	15	17
	1	2	3	4	5	6	7

La valeur **15** qui est le premier élément du tableau **T1** possède **5** éléments qui lui sont strictement inférieurs, donc sa position dans le tableau **T2** est égale à **6** (5+1).

Travail demandé :

- 1) Soit **F** un fichier d'au maximum **100** entiers. Ecrire un algorithme d'une fonction nommée **Comptage (F, a)** permettant de retourner le nombre d'éléments dans le fichier **F** qui sont strictement inférieurs à un entier **a**.
- 2) En se basant sur le principe de tri par comptage décrit précédemment et en utilisant la fonction **Comptage**, écrire un module nommé **Tri_Comptage** qui permet de trier dans un ordre croissant les éléments du fichier **F** dans un deuxième fichier d'entiers **F2**.

NB : On pourra placer les éléments triés dans un tableau puis les transférer dans le fichier **F2**.