Travaux dirigés d'informatique 4

Serie $N°2$

Exercice 1 : (Somme des inverses des carrés d'entiers > 0)

$\forall\, n \in \mathbb{N}^*,$ *on pose :* $S_N = \sum_{n=1}^{N} \frac{1}{n^2}$.

1) *Concevoir et écrire une fonction algorithmique* ***non récursive*** *qui calcule le terme de rang N arbitraire de la suite S_N.* **N.B.** *Essayer d'envisager plusieurs approches.*
2) *On sait que (S_N) est la suite des sommes partielles d'une* ***série numérique convergente****. Concevoir et écrire alors une fonction algorithmique qui somme les termes de cette série jusqu'à un rang N tel que la somme S_N est une approximation de la somme totale de cette série avec une incertitude absolue $< \varepsilon$, avec ε réel > 0 donné, et qui renvoie cette approximation comme résultat.*

Exercice 2 : (Somme des inverses des factorielles - Calcul du nombre e)

*Comme dans l'****Exercice 1****, mais avec :* $\forall\, n \in \mathbb{N},\ S_N = \sum_{n=0}^{N} \frac{1}{n\,!}$.

N.B. *Bien faire attention ici à la problématique posée par la présence de la* ***fonction factorielle****.*

Exercice 3 : (Calcul approchée d'une intégrale par développement en série)

On pose : $I = \int_0^1 \frac{e^x - 1}{x} dx,\ \ A = \sum_{n=1}^{+\infty} \frac{1}{n \cdot (n\,!)}$.

1) *Montrer que $I \in \mathbb{R}$ et $A \in \mathbb{R}$.*
2) *Démontrer que $I = A$.*
3) *Utiliser ce dernier résultat pour construire une méthode numérique permettant de calculer une approximation de I avec une incertitude absolue $< \varepsilon$, avec ε réel > 0 donné, ainsi qu'une approximation de l'erreur absolue associée à cette approximation de I.*
4) *Concevoir et écrire alors une fonction algorithmique mettant cette méthode numérique en œuvre.*

Exercice 4 : (Calcul approchée d'une intégrale par développement en série)

On pose : $I = \int_0^1 \frac{1 - \cos x}{x^2} dx,\ \ S = \sum_{n=1}^{+\infty} \frac{(-1)^{n-1}}{(2n)\,! \cdot (2n-1)}$.

1) *Montrer que* $I \in \mathbb{R}$ *et* $S \in \mathbb{R}$.

2) *Démontrer que* $I = S$.

3) *Utiliser ce dernier résultat pour construire une méthode numérique permettant de calculer une approximation de I avec une incertitude absolue* $< \varepsilon$, *avec* ε *réel* > 0 *donné, ainsi qu'une approximation de l'erreur absolue associée à cette approximation de* I.

4) *Concevoir et écrire alors une fonction algorithmique mettant cette méthode numérique en œuvre.*

Exercice 5 :

On pose : $I = \displaystyle\int_0^{\frac{\pi}{2}} \cos(\sin x)dx$, $A = \displaystyle\sum_{n=0}^{+\infty} \frac{(-1)^n}{4^n(n\,!)^2}$, $B = \displaystyle\sum_{n=0}^{+\infty} \frac{(-1)^n}{(2n)\,!} \cdot J_n$, *où* $J_n = \displaystyle\int_0^{\frac{\pi}{2}} (\sin x)^{2n}dx$.

L'objectif ici est de calculer, avec une précision fixée d'avance, la valeur de l'intégrale I.

1) *Démontrer que :* ***a)*** $A \in \mathbb{R}$; ***b)*** $I = B$; ***c)*** $\forall n \in \mathbb{N}^*$, $(2n) \cdot J_n = (2n-1) \cdot J_{n-1}$.

2) *Déduire l'égalité :* $I = \frac{\pi}{2} \cdot A$.

3) *Utiliser ce dernier résultat pour construire une méthode numérique permettant de calculer une approximation de I avec une incertitude absolue* $< \varepsilon$, *avec* ε *réel* > 0 *donné, ainsi qu'une approximation de l'erreur absolue associée à cette approximation de I.*

4) *Concevoir et écrire alors une fonction algorithmique mettant cette méthode numérique en œuvre.*

Exercice 6 :

On pose : $I = \displaystyle\int_0^{\frac{\pi}{2}} \sin(\cos x)dx$, $A = \displaystyle\sum_{n=0}^{+\infty} (-4)^n \left[\frac{n\,!}{(2n+1)\,!}\right]^2$, $J_n = \displaystyle\int_0^{\frac{\pi}{2}} (\cos x)^{2n+1}dx$,

$B = \displaystyle\sum_{n=0}^{+\infty} \frac{(-1)^n \cdot J_n}{(2n+1)\,!}$.

L'objectif ici est de calculer, avec une précision fixée d'avance, la valeur de l'intégrale I.

1) *Démontrer que :* ***a)*** $A \in \mathbb{R}$; ***b)*** $I = B$; ***c)*** $\forall n \in \mathbb{N}^*$, $(2n+1) \cdot J_n = (2n) \cdot J_{n-1}$.

2) *Déduire l'égalité :* $I = A$.

3) *Utiliser ce dernier résultat pour construire une méthode numérique permettant de calculer une approximation de I avec une incertitude absolue* $< \varepsilon$, *avec* ε *réel* > 0 *donné, ainsi qu'une approximation de l'erreur absolue associée à cette approximation de I.*

4) *Concevoir et écrire alors une fonction algorithmique mettant cette méthode numérique en œuvre.*

Exercice 7 : (Calcul du nombre π - Procédé d'accélération d'Aitken)

On sait que : $\dfrac{\pi}{4} = \displaystyle\sum_{n=0}^{+\infty} \frac{(-1)^n}{(2n+1)}$.

1) *Utiliser ce résultat et le procédé d'accélération d'Aitken(Cf. Chapitre sur la* ***Sommation numérique des séries****) pour construire une méthode numérique permettant de calculer une approximation de*

π avec une incertitude absolue $< \varepsilon$, avec ε réel > 0 donné, ainsi qu'une approximation absolue associée à cette approximation de π.

2) *Concevoir et écrire alors une fonction algorithmique mettant cette méthode numérique en œuvre.*

3) ***Manipulation des grands nombres dans un programme informatique**.*
Dans certaines situations, on a besoin de manipuler des nombres de taille beaucoup plus grande que ceux connus en standard par un langage de programmation. Pour y arriver, la solution consiste alors à **simuler un système de représentation des nombres** du type et de la taille appropriés dans un programme écrit dans ce langage.

<u>Exercice 8 :</u> (Grands entiers dans un programme informatique)

Ici, nous nous intéressons au cas des entiers. Écrire un programme en L.E.A qui permet de manipuler les entiers pouvant atteindre jusqu'à 50 chiffres en base 10. Plus précisément :

1) *Construire un type auxiliaire approprié pour la situation, et qu'on appellera **big_entier.***

2) *Écrire des fonctions algorithmiques booléennes pour traduire respectivement, dans ce cadre, les comparaisons habituelles entre entiers, i.e. $=, \neq, <, >, \leqslant, \geqslant$.*

3) *Écrire des fonctions algorithmiques pour effectuer respectivement, dans ce cadre, les opérations arithmétiques usuelles sur les entiers, i.e. opposé, addition, soustraction, multiplication, reste et quotient de la division euclidienne.*

4) *Écrire une procédure algorithmique de lecture d'un tel entier.*

5) *Écrire une procédure algorithmique d'écriture d'un tel entier.*

N.B. *Cependant, on évitera de bloquer le programme sur la base* 10 *et la taille* 50*. Écrire le programme de telle sorte que ces* 2 *valeurs puissent être facilement changées.*

<u>Exercice 9 :</u> (Grands nombres réels dans un programme informatique)

Ici, nous nous intéressons au cas des nombres réels. Écrire un programme en L.E.A qui simule la manipulation d'un système de représentation de réels-machine en virgule flottante normalisée avec une mantisse de 50 chiffres en base 10, et un exposant à 15 chiffres. Plus précisément :

1) *Construire un type auxiliaire approprié pour la situation, et qu'on appellera **big_réel.***

2) *Écrire des fonctions algorithmiques booléennes pour traduire respectivement, dans ce cadre, les comparaisons habituelles entre réels, i.e. $=, \neq, <, >, \leqslant, \geqslant$.*

3) *Écrire des fonctions algorithmiques pour effectuer respectivement, dans ce cadre, les opérations arithmétiques usuelles sur les réels, i.e. opposé, addition, soustraction, multiplication, division.*

4) *Écrire une procédure algorithmique de lecture d'un tel réel.*

5) *Écrire une procédure algorithmique d'écriture d'un tel réel.*

6) (**Partie subsidiaire**) *Utiliser ce programme pour reprendre l'**Exercice** 7 et calculer π avec la précision maximale qu'il permet. A-t-on obtenu ainsi les* 50 *premières décimales de ce nombre ? Si non, où est le problème et comment le contourner ?*

N.B. ***Cependant, on évitera de bloquer le programme sur la base*** 10 ***et la taille de mantisse*** 50***, et la taille d'exposant*** 15***. Écrire le programme de telle sorte que ces*** 3 ***valeurs puissent être facilement changées.***

<u>Exercice 10 :</u> (Opérations de base sur les polynômes)

Écrire un programme en L.E.A. qui permet de manipuler les polynômes de divers degrés pouvant atteindre 100*. Plus précisément :*

1) *Construire un type auxiliaire approprié pour la situation, et qu'on appellera **polynome.***
2) *Écrire une fonction algorithmique booléenne pour évaluer l'égalité entre* 2 *polynômes.*
3) *Écrire des fonctions algorithmiques pour effectuer respectivement, les opérations arithmétiques usuelles sur les polynômes, i.e. opposé, dérivation, primitive, intégration, addition, soustraction, multiplication, reste et quotient de la division euclidienne.*
4) *Évaluation d'un polynôme en un réel donné.*
5) *Écrire une procédure algorithmique d'écriture d'un tel polynôme.*

N.B. ***Cependant, on évitera de bloquer le programme sur le degrés maximal*** 100***. Écrire le programme de telle sorte que cette valeur puisse être facilement changée.***

<u>Exercice 11 :</u> (Évaluation d'un polynôme pair)

Dans la suite de l'Exercice précédent, concevoir et écrire une fonction algorithmique efficace en un réel donné d'un polynôme dont on sait d'avance qu'il est pair.

<u>Exercice 12 :</u> (Évaluation d'un polynôme impair)

Dans la suite de l'Exercice précédent, concevoir et écrire une fonction algorithmique efficace en un réel donné d'un polynôme dont on sait d'avance qu'il est impair.

<u>Exercice 13 :</u> (Évaluation d'un polynôme de Taylor)

L'objectif ici est l'évaluation algorithmique efficace, en un réel x, du polynôme de Taylor d'ordre n d'une fonction f, centré en un point x_0 :

$$y \;=\; T(x) \;=\; \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k\,!}(x-x_0)^k \;,$$

Où n, x et x_0 sont donnés, ainsi que les coefficients : $b_0 = f(x_0)$, $\;b_1 = f'(x_0)$, $\cdots$, $b_n = f^{(n)}(x_0)$.

1) *On pose* $u_k = \dfrac{(x - x_0)^k}{k\,!}$. *Trouver une relation de récurrence entre* u_k *et* u_{k-1}.

2) *Écrire une fonction algorithmique d'évaluation numérique de* y *qui s'inspire de celui de l'évaluation d'un polynôme* $P(x)$*, écrit sous forme canonique, par calcul des puissances successives de* x.

3) (a) *Trouver l'analogue efficace du* ***schéma de Hörner*** *pour un polynôme de la forme de* $T(x)$.

   (b) *En déduire une fonction algorithmique correspondante pour l'évaluation d'un tel polynôme.*

4) *Comparer l'efficacité numérique relative des* 2 *algorithmes obtenus pour évaluer* $T(x)$.

<u>Exercice 14 :</u> (Polynôme d'interpolation de Lagrange : Forme de Newton)

1) *Écrire une fonction algorithmique en L.E.A. qui prend en entrée :*

   1. *une suite finie de nombre réels distincts :* $x_0, \cdots, x_n$;
   2. *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \cdots, y_n$, *et renvoie comme résultat la suite des différences divisées appropriées pour évaluer le polynôme d'interpolation de Lagrange de* f *relativement aux points d'interpolation* $x_0, \cdots, x_n$.

2) *Écrire une fonction algorithmique en L.E.A. qui prend en entrée :*

   1. *une suite finie de nombre réels distincts :* $x_0, \cdots, x_n$;
   2. *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \cdots, y_n$;
   3. *la suite des différences divisées issu de la fonction algorithmique précédente ;*
   4. *un réel* x *arbitraire,*

   et renvoie comme résultat la valeur en x *du polynôme d'interpolation de Lagrange de* f *relativement aux points d'interpolation* $x_0, \cdots, x_n$ *en s'appuyant sur sa forme de Newton.*

N.B. *Proposer* 2 *versions de cette dernière fonction algorithmique.*

<u>Exercice 15 :</u> (Polynôme d'interpolation de Lagrange : Algorithme d'Aitken)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1) *une suite finie de nombre réels distincts :* $x_0, \cdots, x_n$;
2) *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \cdots, y_n$;
3) *un réel* x *arbitraire,*

   et renvoie comme résultat la valeur en x *du polynôme d'interpolation de Lagrange de* f *relativement aux points d'interpolation* $x_0, \cdots, x_n$, *valeur calculée par l'algorithme d'Aitken.*

<u>Exercice 16 :</u> (Interpolation de Lagrange adaptative par la forme de Newton)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1) *une suite finie de nombre réels distincts :* $a_0, \cdots, a_N$;

2) *la suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0, \cdots, y_N$;*

3) *un réel x arbitraire,*

et qui calcule une valeur approchée de $y = f(x)$:

1. *en calculant successivement les valeurs respectives en x des polynômes d'interpolation de Lagrange de f relativement à x_0, (x_0, x_1), (x_0, x_1, x_2), etc,*
2. *où les points $x_0, x_1, \cdots$ doivent être pris, dans cet ordre, parmi $a_0, \cdots, a_N$; par ordre de proximité décroissante avec x*
3. *et les valeurs des polynômes d'interpolation doivent être calculées en utilisant leur forme de Newton.*

   Ce calcul itératif doit être répété jusqu'à convergence des valeurs calculées avec une incertitude relative $< 10^{-3}$ si c'est possible, et renvoi d'un message d'erreur sinon.

<u>Exercice 17 :</u> (Interpolation de Lagrange par l'algorithme d'Aitken)

*Comme dans **l'Exercice 16**, mais en utilisant plutôt l'algorithme d'Aitken pour l'évaluation en x des polynômes d'interpolation de Lagrange successifs.*

<u>Exercice 18 :</u> (Polynôme d'interpolation d'Hermite : Forme de Newton)

1) *Écrire une fonction algorithme en L.E.A qui prend en entrée :*

1. *Une suite finie de nombres réels distincts : $z_0, \ldots, z_n$;*
2. *La suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0, \ldots, y_m$;*
3. *La suite finie des images correspondantes pour la fonction dérivée f' : $d_0, \ldots, d_m$;*

et renvoie comme résultat la suite des différences divisées approchées pour évaluer le polynôme d'interpolation d'Hermite de f relativement aux couples $(z_0; 1), \ldots, (z_m; 1)$.

2) *Écrire une fonction algorithme en L.E.A. qui prend en entrée :*

1. *Une suite finie de nombres réels distincts : $z_0, \ldots, z_n$;*
2. *La suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0, \ldots, y_m$;*
3. *La suite finie des images correspondantes pour la fonction dérivée f' : $d_0, \ldots, d_m$;*
4. *la suite des différences divisées issu de la fonction algorithme précédente ;*

5. *un réel x arbitraire*,

et renvoie comme résultat la valeur en x du polynôme d'interpolation d'Hermite de f relativement aux points d'interpolation $(z_0;1),\ldots,(z_m;1)$ en s'appuyant sur sa forme de Newton.

N.B. *Proposer 2 versions de cette dernière fonction algorithme.*

Exercice 19 : (Polynôme d'interpolation d'Hermite : Algorithme d'Aitken)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1. *Une suite finie de nombres réels distincts : $z_0,\ \ldots\ ,\ z_n$;*

2. *La suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0,\ldots,y_m$;*

3. *La suite finie des images correspondantes pour la fonction dérivée f' : $d_0,\ldots,d_m$;*

4. *un réel x arbitraire*,

et renvoie comme résultat la valeur en x du polynôme d'interpolation de Lagrange de f relativement aux points d'interpolation $x_0,\ldots,x_n$, valeurs calculée par l'algorithme d'Aitken.

N.B. *Évidemment, il faudra d'abord expliquer l'adaptation de l'algorithme d'Aitken dans ce contexte.*

Exercice 20 : (Interpolation d'Hermite adaptative par la forme de Newton)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée

1. *Une suite finie de nombres réels distincts : $a_0,\ \ldots\ ,\ a_N$;*

2. *La suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0,\ldots,y_N$;*

3. *La suite finie des images correspondantes pour la fonction dérivée f' : $d_0,\ldots,d_N$;*

4. *un réel x arbitraire*,

et qui calcule une valeur approchée de $y = f(x)$:

1. *en calculant successivement les valeurs respectives en x des polynômes d'interpolation d'Hermite de f relativement à x_0, (x_0,x_1), (x_0,x_1,x_2), etc,*

2. *Où les points $x_0,x_1,\ldots$ doivent être pris, dans cet ordre, parmi $a_0,\ldots,a_N$ (après duplication préalable) par ordre de proximité décroissante avec x,*

3. *et les valeurs dezs polynômes d'interpolation doivent être calculées en utilisant leur forme de Newton.*

Ce calcul itératif doit être répété jusqu'a convergence des valeurs calculées avec une incertitude relative $< 10^{-3}$ *si c'est possible, et renvoi d'un message d'erreur sinon.*

Exercice 21 : (Interpolation d'Hermite adaptative par l'algorithme d'Aitken)

*Comme dans l'**Exercice 20**, mais en utilisant plutôt l'algorithme d'Aitken pour l'évaluation en* x *des polynômes d'interpolation d'Hermite successifs.*

Exercice 22 : (Fonction d'approximation affine par morceaux)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1. *une suite finie de nombre réels représentant une subdivision d'un intervalle* $[a, b]$: $a_0, \ldots, a_N$;

2. *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \ldots, y_N$;

3. *un réel* x *arbitraire,*

et renvoie comme résultat la valeur en x *de la fonction d'approximation affine par morceaux de* f *sur* $[a, b]$ *s'appuyant sur la subdivision* $(a_0, \ldots, a_N)$*, si* $x \in [a, b]$*, et renvoie un message d'erreur sinon.*

Exercice 23 : (Fonction d'approximation parabolique par morceaux)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1. *une suite finie de nombre réels représentant une subdivision d'un intervalle* $[a, b]$: $a_0, \ldots, a_N$;

2. *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \ldots, y_N$;

3. *un réel* x *arbitraire,*

et renvoie comme résultat la valeur en x *de la fonction d'approximation parabolique par morceaux de* f *sur* $[a, b]$ *s'appuyant sur la subdivision* $(a_0, \ldots, a_N)$*, si* $x \in [a, b]$*, et renvoie un message d'erreur sinon.*

Exercice 24 : (Fonction d'approximation d'Hermite par morceaux)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1. *une suite finie de nombre réels représentant une subdivision d'un intervalle* $[a, b]$: $a_0, \ldots, a_N$;

2. *la suite finie des images correspondantes pour une certaine fonction* f *(inconnue) :* $y_0, \ldots, y_N$;

3. *la suite finie des images correspondantes pour la fonction dérivée f' (inconnue) : $d_0, \ldots, d_N$;*

4. *un réel x arbitraire,*

et renvoie comme résultat la valeur en x de la fonction d'approximation d'Hermite par morceaux de f sur $[a, b]$ s'appuyant sur ces données, si $x \in [a, b]$, et renvoie un message d'erreur sinon.

Exercice 25 : (Fonction d'approximation spline cubique)

Écrire une fonction algorithmique en L.E.A. qui prend en entrée :

1. *une suite finie de nombre réels représentant une subdivision d'un intervalle $[a, b]$: $a_0, \ldots, a_N$;*

2. *la suite finie des images correspondantes pour une certaine fonction f (inconnue) : $y_0, \ldots, y_N$;*

et construit la fonction d'approximation spline sur $[a, b]$ s'appuyant sur la subdivision $(a_0, \ldots, a_N)$.

Exercice 26 : (Polynômes matrices)

N.B.*Ceci est une partie d'un problème d'examen, d'où une certaine redondance avec d'autres Exercices.*

Dans ce problème :

1. *n et p sont des constantes entières ¿0 fixées* ;

2. *$a_0, \ldots, a_n$ sont des nombres réels* ;

3. *$A_0, \ldots, A_n$ sont les matrices carrées réelles d'ordre p.*

L'objectif ici est alors d'examiner le problème de l'évaluation algorithmique efficace d'un polynôme de degré n d'une variable réelle ou matricielle.

I- Calcul de $P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ en un réel x donné.

1) a) *Écrire l'algorithme d'évaluation numérique de $P(x)$ basé sur le calcul de puissance de x.*

b) *Coût de cet algorithme ?*

2) a) *Écrire un algorithme de calcul de $P(x)$ basé plutôt sur le fait qu'on ait aussi (schéma de HÖrner)* :

$$P(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0.$$

b) *Coût numérique de cet algorithme ?*

3) *Comparer l'efficacité numérique relative de ces 2 algorithmes d'évaluation de $P(x)$.*

II- Quelques fonctions algorithmiques matricielles de base.

1) *Écrire les fonctions algorithmiques respectives :*

   a) *$PROD\_MAT$ faisant le produit de 2 matrices carrées réelles d'ordre p ;*

   b) *ADD_MAT faisant la somme de 2 matrices carrées réelles d'ordre p ;*

   c) *$SCAL\_MAT$ faisant la multiplication d'une matrice carrée réelle d'ordre p par un réel* donné ;

   d) MAT_VEC faisant le produit d'une matrice carrée réelle d'ordre p par un vecteur, dans cet ordre par un vecteur, dans cet ordre ;

   e) *VEC_MAT faisant le produit d'un vecteur par une matrice carrée réelle d'ordre p, dans cet ordre ;*

2) *Evaluer le coût numérique de chacune de ces fonctions algorithmiques.*

***** NOTA : Dans toute la suite de ce problème, les fonctions algorithmiques de II seront supposées écrites, et pourront être appelées partout où le besoin s'en ferait sentir.**

III-Calcul de $F(M) = a_nM^n + a_{n-1}M^{n-1} + \ldots + a_1M + a_0I_p$ **pour** $M \in \mathcal{M}_p(\mathbb{R})$.

*Adapter **I** dans ce cas (sachant que I_p est la matrice d'ordre P).*

IV-Calcul de $G(M) = A_nM^n + A_{n-1}M^{n-1} + \ldots + A_1M + A_0$ **pour** $M \in \mathcal{M}_p(\mathbb{R})$**.**

*Adapter **I** dans ce cas.*

8) ***Tracé de courbes dans un plan***

9) ***Tris de tableaux.***

10) ***Résolution numérique des systèmes d'équations linéaires.***

11) ***Intégration numérique.***

12) ***Résolution numérique d'une équation à une inconnue.***

Exercice 27 : (Tri d'une suite finie par sélection-échange)

***Trier** une suite réelle finie, c'est ré-arranger ses éléments dans l'ordre pré-déterminé. Écrire une fonction algorithme en L.E.A. qui fait ce travail, dans l'ordre croissant, de la manière suivante :*

1) *recherche du plus grand élément dans la suite ;*

2) *positionnement de cet élément là où il faut ;*

3) *répéter la manœuvre de manière appropriée jusqu'à ce que le résultat soit OK.*

N.B. *Proposer une* ***version non récursive et une version récursive*** *de cette fonction algorithmique.*