Les exercices de ce document servent de base à la composition des sujets d'examens.
La réalisation de ce travail est donc une bonne préparation à l'examen.
Bien entendu vous pouvez demander à votre enseignant de regarder votre travail et de vous guider, pour ce faire envoyez un courriel en joignant le code fautif et en donnant votre analyse du problème.
Il n'y a évidement aucun intérêt à lire une quelconque solution…
Prévoir 1h00 au minimum par exercice lorsqu'on est bien entrainé

Exercice 1 :

On vous demande d'écrire un programme C qui résolve le problème suivant. À partir d'un premier ensemble **A** de N1 nombres entiers différents **positifs** donnés et d'un second ensemble **B** de N2 nombres entiers différents positifs donnés, détermine et écrit, s'ils existent, tous les nombres qui sont communs aux deux ensembles. Dans le cas où il n'y aurait pas de nombre commun, le programme l'indiquera par un message.
Les données sont rangées dans une variable dimensionnée dans l'ordre suivant :

{ N, valeur du 1er nombre,… , valeur du N-ième nombre }.

Pour résoudre ce problème vous procéderez **obligatoirement** de la façon suivante :

1. Créez un type de variable dimensionnée pouvant contenir 11 entiers appelé **liste.**
2. Ecrirez une fonction **void saisie(liste A)** qui saisira **N** nombres du premier ensemble pour les ranger dans une variable de type **liste** transmise en paramètre. **N** sera saisi dans la fonction saisie qui empêchera que **N** soit supérieur à 10 en recommençant la saisie tant que **N** est supérieur à 10.
3. Ecrirez une fonction **int cherche(liste A, int x)** qui cherche la présence d'un nombre x dans la variable de type liste **A** et qui retournera :
   - -1 si le nombre n'est pas présent dans le tableau,
   - la valeur de ce nombre sinon.
4. Dans la fonction main(), à l'aide des deux fonctions précédemment écrites, vous créerez deux listes **A** et **B** et procèderez à la saisie de ces deux listes à l'aide de la fonction saisie.
5. Déterminez, à l'aide de la fonction cherche, tous les nombres de la liste **A** appartenant à la liste **B**. Ces nombres seront les rangés dans une troisième liste **C** qui contiendra tous les nombres communs aux deux premières listes.
6. Affichez à l'écran le contenu de la liste C

Indication : L'élément d'indice zéro d'une liste est le nombre de points qu'elle contient

Exercice 2 :

1/ Définissez un type de variable dimensionnée pouvant contenir 10 réels et ayant le nom vect.

2/ Créez une fonction prod_vect() qui pour deux vecteurs x et y donnés renvoie la valeur réelle correspondant au produit scalaire <X,Y>. On rappelle que :

si $X = (x_0 \dots x_{n-1})$ et $Y = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}$ alors le produit scalaire s'exprime par $< X, Y >= \sum_{i=0}^{n-1} x_i y_i$

Cette fonction aura le prototype suivant : float prod_vect(vect x, vect y, int n); x et y étant les vecteurs dont on veut calculer le produit scalaire et n la dimension des vecteurs.

3/ A l'aide de cette fonction calculez les produits scalaires suivants : X = (0 0 1) et Y = (1 0 0) , puis X = (1 - 1 0) et Y = (-1 1 0).

Exercice 3 :

1. Définissez une fonction **fonc()** qui aura **x** un paramètre de type **float** en entrée et retournera un **float**. Cette fonction calculera la valeur suivante : (x-1)*(x-2)*(x-3).

2. Dans la fonction main, calculez des points régulièrement espacés entre 0 et 4, de telle sorte que l'écart entre deux points successifs soit égal à 0,2 et stockez-les dans une variable convenablement dimensionnée **X** de type **vect**. Ce type aura été préalablement créé et pourra contenir 100 **float**.
3. Affichez à l'écran sous forme de deux colonnes séparées par une tabulation la valeurs des **xi** et les valeurs retournées par la fonction **fonc** au point **xi** pour tous les éléments de la variable **X**. Les nombres seront **obligatoirement** affichés sur un minimum de **8** caractères et avec **3** décimales.
4. Dans la fonction principale, déterminez et affichez à l'écran la valeur maximale prise par fonc(X) dans l'intervalle [0, 4].

Exercice 4 :

On se propose d'écrire un programme capable de faire la somme des éléments impairs contenus dans un tableau. Si aucun élément n'est impair, le programme l'affichera par un message.
Pour ce faire vous procéderez **obligatoirement** comme suit :

1. Créez un type de variable capable de contenir 10 valeurs entières. Ce type devra s'appeler **datas**.
2. Dans la fonction main : demandez à l'utilisateur d'entrer N le nombre de valeurs à saisir. La saisie devra se répéter tant que la réponse de l'utilisateur n'est pas comprise dans l'intervalle [2 10].
3. Dans la fonction main : procédez à l'aide d'une boucle à la saisie des **N entiers positifs** dans un vecteur de type datas. **La saisie devra rejeter les valeurs négatives en redemandant la saisie.**
4. Ecrivez une fonction **somme** qui calcule la somme des éléments impairs d'un vecteur transmis en paramètre. Cette fonction aura le prototype suivant : **int somme(datas X, int N)**
   N étant le nombre d'éléments contenus dans la variable X.
5. Dans la fonction main : Affichez la somme des nombres impairs contenus dans X ou un message indiquant qu'il n'y avait pas de nombre impair saisi.

Rappel : x%y renvoie le reste de la division entière de x par y
Indication : 0 est un nombre pair.

Exercice 5 :

Ecrivez un programme en langage C qui, pour une liste de **N** nombres **entiers** *strictement positifs* donnés, détermine et écrit, s'il existe, le plus grand nombre impair situé en une position paire. Dans le cas où ce nombre n'existe pas, on l'indiquera par un message.
Les données sont rangées comme suit : { valeur de N, valeur du 1er nombre, … , valeur du Nième nombre}

1. Créez un type **liste** permettant le stockage de 100 entiers.
2. Ecrivez une fonction **void saisie(liste L)** qui saisisse au clavier **L[0], le nombre de valeurs** de la liste puis les L[i] valeurs de la liste.

***Indication** : veuillez noter que le premier élément de la liste est en position 1 et non 0.*

3. Ecrivez une fonction **int test(liste L)** qui retourne la valeur du plus grand entier impair en position paire de la liste **L**. S'il n'y a pas d'entier impair en position paire, la fonction retournera -1.
4. Ecrivez la fonction **main( )** nécessaire pour testez votre programme avec les listes suivantes :

Exemples :
Cas 1 : {6, 5, 6, 10, **15**, 18, **23**}. Retourne *23* (liste de 6 éléments, 5 est le premier élement).
Cas 2 : {6, 5, 6, 10, 18, 23, 6}. Retourne -1 (*pas ne nombre impair en position paire)*
Cas 3 : {8, 1, 8, 2, **7**, 3, 6, 4, 5}. Retourne *7* (liste de 8 éléments, 1 est le premier élement).

Exercice 6 :

On se propose de calculer la valeur prise par une fonction sur N points **régulièrement** espacés.

1. Dans la fonction **main( )**, saisissez le nombre **N** avec un message d'invite. Le programme interdira la saisie de valeurs hors de l'intervalle [2 ; 99].
2. Définissez **tab** un type de variable à une dimension, pouvant contenir au maximum 100 valeurs **réelles**.
3. Créez deux variables **X** et **Y** de type **tab**. Faites remplir par votre programme la variable **X** de valeurs régulièrement réparties entre -2 et 2 sur **N** points **sans saisir au clavier les valeurs une à une**.

➡ *Il ne s'agit pas ici de faire saisir les N valeurs X[i] mais de les calculer !*

***Indication** : Calculez h le pas, puis tous les **X[i]** sachant que X[0] = -2.*

4. Calculez les valeurs prises par la fonction y = F(x) définie par $F(x) = 7x^2 - x + 3$ où x est l'élément X[i] du vecteur **X** calculé au point 3, et y l'élément Y[i] du vecteur **Y** défini ci-dessus.
5. Ecrire une fonction appelée **sommezero( )** qui calcule la somme des valeurs supérieures à zéro d'une variable dimensionnée et qui renvoie le résultat sous forme de **float**.

***Indication** : vous devez déterminer quels sont les paramètres à transmettre à **sommezero**, ainsi que son type.*

6. Dans la fonction **main( )**, à l'aide de la fonction **sommezero**, calculez et affichez la somme des valeurs strictement positives prise par la fonction F(x) dans l'intervalle [-2 ; 2].

Exercice 7 :

1. Définissez un nouveau type **mat** qui correspond à une matrice d'éléments *entiers* de dimension **5 par 5**.
2. Dans le programme principal, créez une matrice **M** de type **mat**. Remplissez, à l'aide de boucles, la matrice de telle sorte que l'on ait : **M[i][j]** = 3***i** - 2***j**.
3. Créez une fonction appelée **max_mat** qui détermine la plus grande valeur d'une matrice *d'entiers* et en retourne la valeur.
4. Créez une fonction appelée **trace_mat** qui calcule la trace des éléments d'une matrice d'entiers et renvoie la valeur.

➡ *On rappelle que la trace d'une matrice est la somme des éléments sur la diagonale principale.*

5. Calculez et affichez la trace de la matrice calculée à la question 2.
6. Affichez la plus grande des valeurs de la matrice.

Exercice 8 :

1. Ecrivez une fonction typée nommée **carre** de type **float** permettant de renvoyer le carré d'un **entier** transmis en paramètre.
2. Ecrivez une fonction typée nommée **cube** de type **float** permettant de renvoyer le cube d'un **entier** transmis en paramètre.

➡ *On se propose de calculer, en utilisant les deux fonctions écrites ci-dessus, la somme des carrés des entiers impairs ou des cubes des entiers pairs de l'intervalle [1 , **N**].*

3. Ecrivez une fonction **int sommeI(int p, int N)**. **p** peut valoir 2 ou 3 à l'exclusion de toute autre valeur.
   Si **p** vaut 2, **sommeI** calcule et retourne la somme des carrés des entiers impairs de [1, **N**].
   Si **p** vaut 3 **sommeI** calcule et retourne la somme des cubes entiers pairs de [1 , **N**].
   - **N** et **p** seront transmis en paramètres à la fonction, *ils doivent être saisis dans la fonction principale*.
   - La valeur « 0 » n'est pas comptée comme un entier pair.
   - Vous devez **impérativement** utiliser dans la fonction **sommeI** les fonctions que vous avez créées lors des questions 1 et 2 sous peine de nullité de votre réponse.
   - La fonction **sommeI** doit renvoyer la valeur -99 si **N** ou **p** ne permettaient pas le calcul des sommes.
4. Dans la fonction **main( )**, calculez puis affichez la somme des carrés des entiers impairs positifs inférieurs à 10, puis celle du cube des entiers pairs positifs inférieurs à 20.

Exercice 9 :

Soit les n (< 100) entiers (donnés par ordre croissant) suivants. { -2 ; 7; 11; 13; 19; 23; 31; 37; 41; 43}

1. Créez un type vect permettant de stocker 100 entiers.
2. Dans la fonction main, écrivez le code permettant la saisie des éléments de la liste ci-dessus dans une variable de type **vect A** à l'aide d'une boucle. Lors de la saisie, vous procéderez à un affichage de confirmation systématique, sous la ligne de saisie, comprenant le message :
   "Valeur saisie X" ou X sera remplacé par la valeur saisie.
   ```
   Exemple :
          Entrez une valeur : 11
          Valeur saisie  : 11
          Entrez une valeur : _
   ```
3. Ecrivez une fonction **moyenne** qui retourne la valeur moyenne des éléments d'une variable de type **vect**.
4. Calculez dans la fonction main la valeur moyenne **a_moy** des éléments du vecteur **A** à l'aide de la fonction **moyenne**.
5. Dans la fonction main, recherchez la valeur du vecteur **A** la plus proche (mais supérieure) à la valeur **a_moy** et vous afficherez à l'écran son indice et sa valeur.

Indication : quel est le type du résultat de la fonction moyenne ?

Exercice 10 :

On se propose de déterminer le nombre de valeurs d'une liste de **N** entiers qui sont multiples de trois et de calculer la moyenne de ces valeurs. Pour ce faire vous procéderez comme suit :

1. Créez un type de variable appelé **vect** correspondant à une variable à une dimension de 100 éléments entiers.
2. Dans la fonction **main( )** stockez dans une variable **X** de type **vect** les différentes valeurs de la liste suivante : { 13, 5, 98, 45, 21, 58, 43, 26, 75, 42 }.
   - Ces valeurs devront être saisies au clavier par l'utilisateur ainsi que **N** le nombre de valeurs à saisir.
   - Lors de la saisie à l'aide d'une boucle, vous procéderez à un affichage de confirmation systématique sous la ligne de saisie, comprenant le message : "Valeur saisie X" ou X sera remplacé par la valeur saisie.

     Exemple :
     ```
     Entrez une valeur : 13
     Valeur saisie  : 13
     Entrez une valeur : _
     ```
3. Créez une variable **Y** de type **vect** que vous remplirez avec la règle suivante :
   - **Y**[i] = 0 si **X**[i] n'est pas multiple de trois,
   - **Y**[i] = 1 sinon.
4. En utilisant le vecteur **Y,** affichez les indices et valeurs des éléments de **X** multiples de 3 sous forme d'une liste à deux colonnes.
5. Affichez le nombre de valeurs de **X** qui sont multiples de trois.
6. Calculez et affichez la valeur moyenne des nombres de **X** qui sont multiples de trois.

Exercice 11 :

Soit la fonction F(x) définie par F(x) = y = 3*x + 5

Le programme que vous écrirez devra stocker dans deux vecteurs X et Y (judicieusement créés et définis[1]) les différentes valeurs prises par x et y=F(x), lorsque x varie de x_{min} = - 2 à x_{max} = 8 avec un pas h = 1. C'est à dire : calculez les x_i (X[i]) et y_i (Y[i]). Vous procéderez **obligatoirement** comme suit :

1. Ecrivez un programme qui demandera à saisir au clavier x_{min} , x_{max} et h.
   Tous les autres calculs devront se faire dans le programme.
2. À l'aide des valeurs saisies, calculez N le nombre de points, puis calculez les valeurs prises par les éléments du X[i] vecteur X dans l'intervalle [xmin ; xmax].
3. Ecrivez une fonction qui aura obligatoirement le prototype suivant : **int fonc(int x)** ; la fonction **fonc** doit calculer et retourner la valeur de F(x) en x.
4. Dans la fonction main, calculez les valeurs prises par le vecteur Y : Y[i] = F(X[i]) à l'aide de la fonction **fonc**.
5. Écrivez une fonction **moyenne** qui calcule et retourne la moyenne des éléments d'un vecteur.
6. Calculez et affichez la moyenne du vecteur Y : ymoy1.
7. Calculez et affichez la moyenne de X : xmoy. Calculez et affichez ymoy2 = F(xmoy). Comparerez les valeurs ymoy2 et ymoy1.

Indication : quel est le type du résultat de la fonction moyenne ?

Exercice 12 :

1. Créez un nouveau type de variable appelé **mat** correspondant à une matrice carrée de **5** lignes par **5** colonnes d'entiers.
2. Ecrivez une fonction appelée **affiche** qui affiche les éléments d'une variable de type **mat** sous forme matricielle. Chaque élément occupera au minimum 4 caractères à l'écran.
3. Dans la fonction principale, créez puis emplissez une matrice **A** telle que chacun de ses éléments a_j^i soit égal au produit des indice de ligne i et colonne j.

Indication : C'est la notation mathématique et non informatique qui est utilisée ici le premier élément est donc 1, et non 0.

4. Afficher la matrice **A**, vous devez avoir :

```
 1   2   3   4   5
 2   4   6   8  10
 3   6   9  12  15
 4   8  12  16  20
 5  10  15  20  25
```

5. Dans la fonction principale, créez une matrice **B** de type **mat** contenant les éléments pairs de la matrice **A**, les autres éléments étant nuls. Affichez la matrice **B** en utilisant la fonction affiche.
   Nota Bene : Les éléments de **B** ne seront pas recalculés mais évalués à partir des éléments de la matrice **A**.
6. Dans la fonction principale, créez une matrice **C** de type **mat** contenant les éléments impairs de la matrice **A**, les autres éléments étant nuls. Affichez la matrice **C** en utilisant la fonction affiche.
   Nota Bene : Les éléments de **C** ne seront pas recalculés mais évalués à partir des éléments de la matrice **A**.
7. Dans la fonction principale calculez une matrice **D** de type **mat**, somme des deux matrices **B** et **C**. Affichez la matrice **D**.
8. Vérifiez que l'on a bien égalité des matrices **A** et **D** en changeant le signe de chaque élément de **D**, puis en calculant et affichant la somme **A**+**D**.

[1] Il est conseillé de créer un type mais ce n'est pas obligatoire.

Exercice 13 :

On se propose de représenter à l'écran le dessin utilisant le motif suivant : "/_\ ", de façon à produire des graphiques de k lignes (k étant saisi au clavier et compris dans l'intervalle [1 ; 6].

```
Combien de lignes : 4
/_\_1__2__3__4__5_/_\
.../_\_1__2__3_/_\...
....../_\_1_/_\......
.........../_\.........
```

```
Combien de lignes : 6
/_\_1__2__3__4__5__6__7__8__9_/_\
.../_\_1__2__3__4__5__6__7_/_\...
....../_\_1__2__3__4__5_/_\......
.........../_\_1__2__3_/_\.........
................/_\_1_/_\..............
.................../_\.................
```

Pour ce faire, vous procéderez en respectant les étapes suivantes :

1/ Ecrivez une fonction appelée **motif** qui affiche à l'écran la chaine de caractères "/_\".

Indication : *Attention le caractère '\' a une signification particulière en langage C.*

2/ Ecrivez une fonction ayant le prototype : **void ligne(int n)**, affichant la ligne : "_1_..._n_ " à l'aide d'une boucle affichant le motif _k_ ou k varie de 1 à n.

3/ Affichez la première ligne dans le cas où l'on voudrait une figure à 6 lignes.

Indications : *Combien de lignes avec la fonction ligne devrez-vous afficher avec la fonction ligne pour la figure à 4 lignes ? Même question pour la figure à 6 lignes ? Quelle est la parité de n, paramètre de la fonction ligne ? Comment change n d'une ligne affichée à l'autre ?*

4/ Ecrivez une fonction ayant le prototype : **void point(int n**). Cette fonction affichera n fois le motif "...".

5 / A l'aide des fonctions **motif**, **point**, et **ligne** affichez les n-1 premières lignes de la figure.

6/ Complétez la figure avec la dernière ligne.

7/ Complétez le programme pour n'autoriser que la saisie d'un nombre de ligne compris dans l'intervalle autorisé. En cas de saisie erronée le programme affichera un message rappelant les bornes à respecter.

Exercice 14 :

1. Définissez un type global appelé **vect** permettant le stockage de 50 variables de type **float**.
2. Ecrivez une fonction permettant de calculer les **n** premiers termes de la suite définie par : $x_{k+1} = x_k + x_{k-1}$. Le prototype de cette fonction sera : **void fibo(float x0, float x1, int n, vect x)**.
3. Calculez les n premiers termes de cette série en prenant x_0=1, x_1=1 et n=10 à l'aide de la fonction écrite ci-dessus. Les valeurs $\mathbf{x_0}$, $\mathbf{x_1}$ et **n** seront saisies dans le programme principal. Vous devez trouver $x_{10} = 89$. Aucune saisie n'est demandée.
4. Affichez à l'écran les valeurs trouvées sous la forme de deux colonnes. La première colonne contiendra l'indice **k** du terme x_k et la seconde sa valeur.
5. Ecrivez une fonction qui détermine et retourne le nombre de valeurs paires contenues dans une liste **L**. Cette fonction devra, de plus, calculer la somme des nombres pairs contenus dans **L** ainsi que la moyenne de ces nombres pairs. Le prototype de la fonction sera : **int pair(vect L, int n).**
   L contiendra la liste à traiter, et **n** le numéro du dernier élément de cette liste.
   La somme des nombres pairs sera stockée l'élément n+1 de la variable **L**, la moyenne dans l'élément d'indice **n+2**.

***Indication** : Attention, les éléments de **L** sont des **float**.*

6. Faite saisir à l'utilisateur les deux valeurs initialisant la suite et ***n*** le nombre de termes à calculer de telle sorte que le programme n'autorise la saisie de ***n*** que s'il est compris dans l'intervalle allant de 3 à 30 (bornes incluses). En cas de saisie erronée le programme affichera un message rappelant les bornes à respecter et effectuera à nouveau la saisie.
7. À l'aide des questions précédentes, déterminez le nombre de valeurs paires dans les 25 premiers termes de la suite de Fibonacci initialisée avec les valeurs 2 et 3.
8. Affichez la somme de ces termes ainsi que leur moyenne.
   Indiquez les valeurs trouvées aux questions 7 et 8 en commentaire dans votre programme.

Exercice 15 :

On se propose de réaliser un algorithme de tri sur index.

1/ Créez un type de variable appelé **Liste** pouvant contenir 50 entiers.

2/ Dans la fonction main(), créez une variable de type **Liste** appelée **datas** qui contient les éléments suivants : { 3, 8, 6, 4,-2, 7, 0, 1, -4, -1}.

3/ Ecrivez une fonction qui détermine l'indice du plus petit élément d'une **Liste** transmise en paramètre. Le prototype de la fonction sera : **int cal_min(Liste x , int N)**.

N contient le nombre d'éléments de la **Liste x**.

4/ Créez une variable de type **Liste** appelée **copie**, puis copiez le contenu de la variable **datas** dans la variable **copie** à l'aide d'une boucle dans la fonction main().

5/ Créez une variable de type **Liste** appelée **index.**

6/ A l'aide de l'algorithme suivant, triez la liste **copie**.

a) déterminez **M** l'indice du plus petit élément de **copie** à l'aide de la fonction **cal_min**.

b) Stockez **M** dans l'élément **k** de la variable **index.**

c) Remplacez la valeur du plus petit élément de **copie** par 999.

d) Recommencez au point a) tant que tous les points de la variable **copie** ne contiennent pas 999.

Indications : k est un indice de boucle. On parcours la variable **index** de la première à la dernière. **index** aura forcément autant d'éléments que **datas**.

7/ En utilisant la variable **index** affichez les valeurs de **datas** par ordre croissant.

8/ En utilisant la variable **index** affichez les valeurs de **datas** par ordre décroissant.

Le format d'affichage des données des questions 7 et 8 sera du type :

indice : 8 valeur : -4

indice : 4 valeur : -2 etc.