

**Questionnaire
examen final
+corrigé**

INF1005C

Sigle du cours

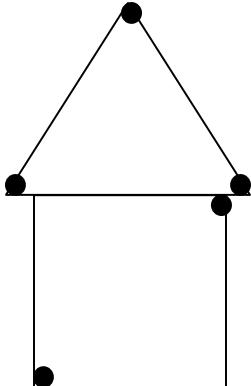
Identification de l'étudiant(e)

| | | |
|--------------------|--------------------|-----------------|
| Nom : | Prénom : | |
| Signature : | Matricule : | Groupe : |

| Sigle et titre du cours | | Groupe | Trimestre |
|---|---|--|-------------------|
| INF1005C – Programmation procédurale | | Tous | 20091 |
| Professeur | | Local | Téléphone |
| Martine Bellaïche, responsable et Jean-Charles Bernard | | 3414 | 4679/5193 |
| Jour | Date | Durée | Heures |
| Jeudi | 23 avril 2009 | 2h | 9h30-12h00 |
| Documentation | | Calculatrice | |
| <input checked="" type="checkbox"/> Aucune | <input checked="" type="checkbox"/> Aucune | Les téléphones cellulaires, agendas électroniques ou télefavertisseurs sont interdits. | |
| <input type="checkbox"/> Toute | <input type="checkbox"/> Tout | | |
| <input checked="" type="checkbox"/> Voir directives particulières | <input type="checkbox"/> Non programmable | | |
| Directives particulières | | | |
| <ul style="list-style-type: none"> <input type="checkbox"/> Ne recopiez pas les déclarations ou instructions déjà fournies dans le questionnaire. <input type="checkbox"/> Vous n'avez pas à écrire de commentaires, ni sur-têtes. <input type="checkbox"/> On ne répondra à aucune question. En cas de doute, veuillez faire vos suppositions et les écrire sur le cahier d'examen. | | | |
| <i>Bonne chance à tous!</i> | | | |
| Important | Cet examen contient 5 questions sur un total de 5 pages (excluant cette page) | | |
| | La pondération de cet examen est de 40 % | | |
| Vous devez répondre sur : <input type="checkbox"/> le questionnaire <input checked="" type="checkbox"/> le cahier <input type="checkbox"/> les deux | | | |
| Vous devez remettre le questionnaire : <input type="checkbox"/> oui <input checked="" type="checkbox"/> non | | | |

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

1. Questions générales (4 points)



- 1.1. On désire créer un enregistrement pour représenter une maison. Donnez les définitions des enregistrements, sachant qu'une maison est composée d'un rectangle et d'un triangle, qu'un rectangle est composé de 2 points opposés et qu'un triangle de 3 points.
- 1.2. Écrivez les déclarations et les instructions permettant d'ouvrir un fichier binaire donnee.bin.
- 1.3. Expliquez le but de la fonction `sizeof()`.
- 1.4. Soit l'enregistrement suivant :

```
struct patient
{
    string nom;
    int age;
};
```

 - 1.4.1. Donnez la déclaration d'une variable de type pointeur à `patient`.
 - 1.4.2. Donnez l'instruction permettant d'allouer dynamiquement de l'espace mémoire de type `patient` à ce pointeur.
 - 1.4.3. Donnez les instructions pour initialiser cet espace mémoire aux valeurs {"tremblay",45}.

Solution question 1

1.

```
struct point
{
    float x,y;
};

struct carre
{
    point P1,P2;
};

struct triangle
{
    point P1,P2,P3;
};

struct maison
{
    triangle toit;
    carre batisse;
};
```

2. ifstream fichier;
fichier.open("donnee.bin",ios::binary);

3. sizeof() retourne le nombre d'octets d'un type

4.

```
patient * nouveau;
nouveau = new patient;
nouveau->nom ="tremblay";
nouveau->age = 45;
```

2. Erreurs de programmation (4 points)

```

1. #include <iostream>
2. #include <fstream>
3. using namespace std;
4. void main()
5. {
6.     ofstream lecture;
7.     lecture.open("note.txt");
8.     int noteA , noteB ,
9.         noteC, noteD,
10.        noteF, note;
11.     if (lecture.fail())
12.     {
13.         lecture << note;
14.         while(!lecture.eof());
15.         {
16.             if ( note >= 80 )
17.                 noteA++;
18.             if (note >= 70)
19.                 noteB++;
20.             if (note >=60)
21.                 noteC++;
22.             if (note >= 50)
23.                 noteD++;
24.             else
25.                 noteF++;
26.             lecture << note;
27.         }
28.         cout << " nombre de A " << noteA << endl;
29.         cout << " nombre de B " << noteB << endl;
30.         cout << " nombre de C " << noteC << endl;
31.         cout << " nombre de D " << noteD << endl;
32.         cout << " nombre de F " << noteF << endl;
33.     }
34.     cout << "fichier inexistant" << endl;
35. }
```

Le programme ci-dessus compte le nombre d'occurrences pour les notes A, B, C, D et F selon les intervalles suivants :

| | |
|---------------|-----------|
| [80 et plus] | Lettre A. |
| [70, 79] | Lettre B. |
| [60,69] | Lettre C. |
| [50,59] | Lettre D. |
| [49 et moins] | Lettre F. |

Ce programme comporte des erreurs. Indiquez et corrigez les lignes afin que ce programme soit fonctionnel et affiche les bons calculs.

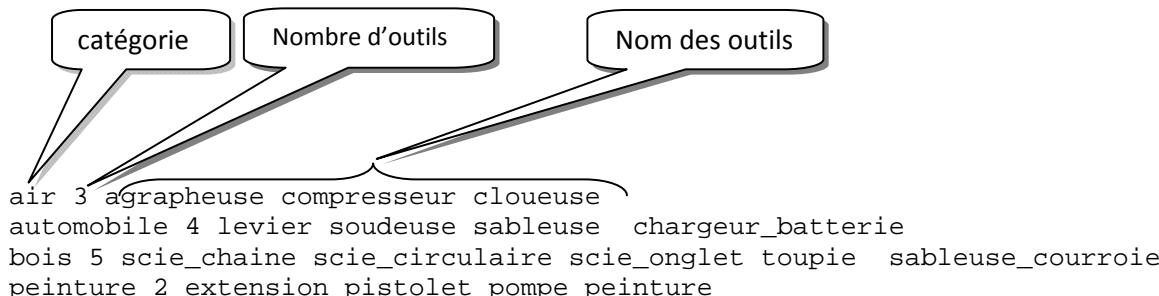
Solution Question 2

```
#include <iostream>
#include <fstream>
using namespace std;

void main()
{
    ifstream lecture;
    lecture.open("note.txt");
    int noteA = 0, noteB = 0,
        noteC = 0, noteD = 0,
        noteF = 0, note;
    if (!lecture.fail())
    {
        lecture >> note;
        while(!lecture.eof())
        {
            if (note >= 80)
                noteA++;
            else if (note >= 70)
                noteB++;
            else if (note >= 60)
                noteC++;
            else if (note >= 50)
                noteD++;
            else
                noteF++;
            lecture >> note;
        }
        cout << " nombre de A " << noteA << endl;
        cout << " nombre de B " << noteB << endl;
        cout << " nombre de C " << noteC << endl;
        cout << " nombre de D " << noteD << endl;
        cout << " nombre de F " << noteF << endl;
        lecture.close();
    }
    else
        cout << "fichier inexistant" << endl;
}
```

3. Fonctions (3 points)

Une compagnie de location d'outils conserve dans un fichier l'inventaire de ses produits. Sur chaque ligne du fichier `outils.txt`, on retrouve le nom de la catégorie, le nombre d'outils de la catégorie et le nom des outils de cette catégorie. On ne connaît pas le nombre de catégories contenues dans le fichier. Voici un exemple de fichier :



La compagnie désire créer un petit programme lui permettant d'effectuer les opérations suivantes : afficher le contenu de son inventaire, ajouter un outil dans une catégorie. Lorsque le programme se termine, on modifiera le contenu du fichier s'il y a eu des changements dans les différentes catégories.

- 3.1. **Sans écrire le programme principal** et en utilisant les pointeurs, donnez la définition des structures et les déclarations des variables nécessaires à la réalisation de ce programme.
- 3.2. **Sans écrire l'implémentation des fonctions**, identifiez toutes les fonctions nécessaires à la réalisation de ce programme. Donnez l'en-tête de toutes les fonctions que vous avez identifiées, à savoir le type de retour, le nom de la fonction, le nombre et le type des paramètres.

Solution Question 3

```

struct categorie
{
    string nom;
    int nombreOutils;
    string * listeOutils;
};

struct tousCategorie
{
    int nombreCategorie;
    categorie * listeCategorie;
};

void main()
{
    tousCategorie Compagnie;
}
bool lectureFichier(tousCategorie & maCompagnie);
void AfficherContenu(tousCategorie maCompagnie);
bool ajouterOutil(tousCategorie & maCompagnie);
void terminer(tousCategorie maCompagnie);

```

4. Allocation dynamique (4 Points)

La structure suivante permet de stocker les titres des CDs.

```
struct tousCD
{
    string* listeCD;
    int nombreCD;
    int capaciteListe;
};
```

Le champ `listeCD` est un pointeur d'un espace mémoire alloué dynamiquement dont la capacité est contenue dans le champ `capaciteListe`. Ce champ `listeCD` contient la liste des titres des CDs dont le nombre est contenu dans le champ `nombreCD`. Si l'on suppose qu'une variable de type `tousCD` contient toute la liste des CDs, écrivez la fonction qui nous permet d'ajouter un titre de CD dans cette variable. S'il n'y a pas assez d'espace mémoire, on doublera la capacité de la mémoire allouée dynamiquement.

Solution Question 4

```
bool ajouterCD(tousCD & mesCD)
{
    bool alloue = true;
    string nouveauCD;
    if (mesCD.nombreCD == mesCD.capaciteListe)
    {
        string * temp = new string [mesCD.capaciteListe*2];
        if (temp != 0)
        {
            for (int i = 0; i < mesCD.nombreCD; i++)
                temp[i] = mesCD.listeCD[i];
            if (mesCD.listeCD!=0)
                delete[] mesCD.listeCD;
            mesCD.capaciteListe = mesCD.capaciteListe*2;
            mesCD.listeCD = temp;
        }
        else
            alloue = false;
    }
    cout << "nouveau CD ";
    cin >> nouveauCD;
    mesCD.listeCD[mesCD.nombreCD++] = nouveauCD;
    return alloue;
}
```

5. Général (5 points)

Un carré magique de lettres est une forme de mots croisés disposés en carré, ne comportant pas de case noire et constitué de mots valides. Les mêmes mots peuvent être lus dans les deux sens, horizontalement et verticalement (source Wikipedia). Voici un exemple d'un carré magique de mots de 4 lettres.

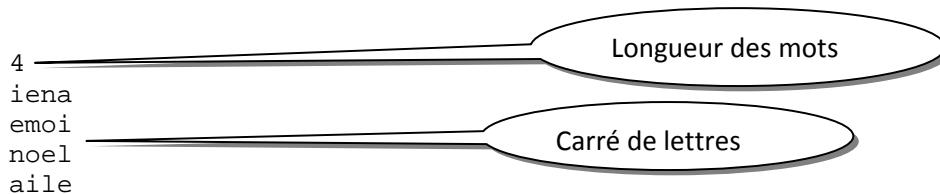
Le contenu d'un carré de lettres est stocké dans une variable de type

```
struct carre
{
    int longueurMot;
    string listeMotHorizontal[20];
};
```

| | | | |
|---|---|---|---|
| I | E | N | A |
| E | M | O | I |
| N | O | E | L |
| A | I | L | E |

Le carré de lettres peut avoir au maximum 20 mots de 20 lettres. Le champ `longueurMot` représente le nombre de lettres du carré. Le champ `listeMotHorizontal` est le contenu du carré de lettres.

- 5.1. Écrivez la fonction `bool lireCarre(carre& unCarre)` qui lit le fichier `carre.txt` et stocke le nombre de lettres et le contenu du carré dans le paramètre `unCarre`. La fonction retourne vraie si le fichier a pu être ouvert, sinon la valeur fausse. On suppose que le **fichier n'est pas vide**. Le fichier a le format suivant :



- 5.2. Écrivez la fonction `bool verifieMagique(carre unCarre)` qui vérifie si le champ `listeMotHorizontal` du paramètre `unCarre` est un carré magique de lettres.

Solution question 5

```
bool lireCarre(carre &unCarre)
{
    ifstream lecture;
    lecture.open( "carre.txt" );
    if (!lecture.fail())
    {
        lecture >> unCarre.longueurMot;
        for (int i = 0; i < unCarre.longueurMot ;i++)
            lecture >> unCarre.listeMotHorizontal[i];
        lecture.close();
        return true;
    }
    else
        return false;
}

bool verifieMagique(carre unCarre)
{
    bool magique = true;
    for (int i = 0 ; (i < unCarre.longueurMot) && magique; i++)
        for (int j =0 ; (j < unCarre.longueurMot)&& magique; j++)
            if (unCarre.listeMotHorizontal[i][j]!=
                unCarre.listeMotHorizontal[j][i])
                {
                    magique = false;
                }
    return magique;
}
```