POLYTECHNIQUE MONTRÉAL

# Questionnaire examen intra
# 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** | **20131** |
| ***Professeur*** | ***Local*** | ***Téléphone*** |
| **Martine Bellaïche** | **M-3414** | **4679** |

| *Jour* | *Date* | *Durée* | *Heures* |
|---|---|---|---|
| **Vendredi** | **1er mars 2013** | **1 h 50** | **8h30-10h20** |

| *Documentation* | *Calculatrice* | |
|---|---|---|
| ☒ Aucune | ☒ Aucune | **Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.** |
| ☐ Toute | ☐ Toutes | |
| ☒ Voir directives particulières | ☐ Non programmable | |

***Directives particulières***

- Ne recopiez pas les déclarations, ni les instructions déjà fournies dans le questionnaire.
- Vous n'avez pas à écrire de commentaires, ni d'en-têtes, ni les includes.
- **On ne répondra à aucune question. En cas de doute, veuillez faire vos suppositions et les écrire sur le cahier d'examen.**

***Important***

Cet examen contient **4** questions sur un total de **4** pages **(excluant cette page)**

La pondération de cet examen est de **25** **%**

Vous devez répondre sur : ☐ le questionnaire ☒ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☐ oui ☒ non

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

# Question 1 (4 points)

Le programme suivant permet de lire le contenu d'un fichier et de stocker chaque ligne du fichier dans un tableau de type string. Le nom du fichier est fourni par l'usager. Remplacer les éléments absents identifiés par *** du programme ci-dessous. Ne pas recopier tout le programme, indiquer seulement le numéro des lignes et le contenu des éléments absents.

```
1. int main()
2. {   const int MAX = 100;
3.     string  nomFichier;
4.     fstream  fichierLu;
5.     string phrase[MAX];
6.     bool estOuvert ***;
7.     bool continuerLecture ***;
8.     int nombreLignes ;
9.     while (!estOuvert)
10.    {
11.        cout << "  veuillez donner le nom du fichier ";
12.        cin  >> nomFichier;
13.        fichierLu.open(nomFichier, ***);
14.        if (!fichierLu.***)
15.           estOuvert = ***;
16.     }
17.     nombreLignes = 0;
18.     while(continuerLecture)
19.     {
20.        getline(fichierLu,phrase[nombreLignes]);
21.        if (fichierLu.***())
22.           continuerLecture = ***;
23.        else
24.           ***;
25.     }
26.     // la fonction FormaterPourConsole
27.     //permet d'afficher les accents
28.     // dans la fenêtre console.
29.     for ( int i = 0 ; i < nombreLignes ; i++)
30.        cout << FormaterPourConsole( ***) <<endl;
31.     fichierLu.***;
32.     return 0;
33.}
```

## Question 2 (4 points)

Le jeu du pendu consiste à trouver un mot secret en donnant des lettres une par une. Si les lettres se trouvent dans le mot, on les place à la bonne position. Le programme lit le mot secret d'un fichier et ensuite

1. demande à l'usager de donner une par une des lettres, si une lettre se trouve dans le mot secret, on la place au bon endroit dans le mot;
2. affiche le nombre d'essais (c'est-à-dire le nombre de lettres) effectués avant que l'usager trouve le mot secret.

Écrire le programme qui implémente le jeu du pendu en respectant les étapes données en commentaires (ne pas faire l'étape 2) et l'exemple d'exécution.

| | |
|---|---|
| *tolower(int argument)* | Convertit un caractère en minuscule. |
| *toupper(int argument)* | Convertit un caractère en majuscule. |

```
int main()
{
// 1.déclarations des variables
// motSecret variable lue dans le fichier
// mot variable de l'usager
  string  motSecret, mot;

// 2. lecture du mot secret dans un fichier Ne pas le faire

// 3. mettre tous les lettres en minuscules du mot secret

// 4. mettre '_' dans les lettres de la variable mot

// 5. boucle  tant que l'on n'a pas trouvé le mot
// dans la boucle  afficher les lettres trouvées
// et les caractères _ pour les lettres manquantes
// déterminer le nombre d'essais
// demander à l'usager une lettre et trouver si elle
// est dans le mot secret, si oui, placer la lettre
// dans le mot

// afficher le nombre d'essais
 return 0;
 }
```

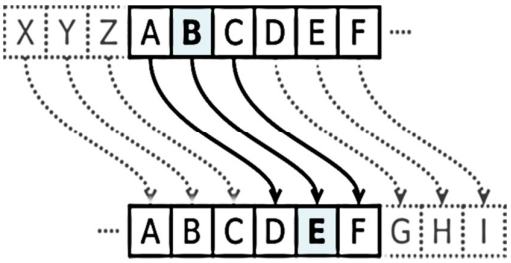
Voici un exemple d'exécution :

```
_ _ _ _ _ _ _ _ _ _
donner une lettre a
_ _ a _ _ _ _ _ _ _
donner une lettre j
_ _ a _ _ _ _ _ _ _
donner une lettre i
_ _ a _ _ i _ _ _ _
donner une lettre m
_ _ a m _ i _ _ _ _
donner une lettre k
_ _ a m _ i _ _ _ _
donner une lettre c
c _ a m _ i _ _ _ _
donner une lettre p
c _ a m p i _ _ _ _
donner une lettre s
c _ a m p i _ _ _ _
donner une lettre z
c _ a m p i _ _ _ _
donner une lettre h
c h a m p i _ _ _ _
donner une lettre g
c h a m p i g _ _ _
donner une lettre n
c h a m p i g n _ n
donner une lettre o
 bravo vous avez trouve le mot en 13 essai
Appuyez sur une touche pour continuer...
```

# Question 3 (5 points)

« En cryptographie, le **chiffrement par décalage à droite**, aussi connu comme le **chiffre de César**, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »).

Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres, on reprend au début. Par exemple avec un décalage de 3 vers la droite, `A` est remplacé par `D`, `B` devient `E`, et ainsi jusqu'à `W` qui devient `Z`, puis `X` devient `A` etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la *clé* du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles. » (source : Wikipédia ).




Par exemple dans l'image, il y a un décalage de 3 caractères, donc `B` devient `E` dans le texte chiffré.

En résumé, si `position` est la position de la lettre du message en clair et si `decalage` (entier positif) est la clé de chiffrement, alors la position de la lettre du message chiffré est `(position + decalage) %26.`

En fait, on désire trouver le décalage ou la clé de chiffrement de la méthode de chiffrement de César, ayant à notre connaissance le message en clair et le message chiffré,

Soient les variables suivantes :

```
string  lettre ="abcdefghijklmnopqrstuvwxyz";
string phraseLisible;
string phraseCryptee;
int decalage, position;
```

En supposant que `phraseLisible` et `phraseCryptee` sont lues dans un fichier (ne pas faire), écrire un programme qui trouve et affiche le décalage ou la clé de chiffrement.

# Question 4 (7 points)

Ayant une liste de noms correspondant à des fichiers et une liste d'extensions, le programme vérifie l'existence des fichiers et en compte leur nombre. Voici un exemple du contenu des fichiers :

| Fichier de liste de noms | Fichier de liste d'extensions |
|---|---|
| controle | doc |
| dessin | xls |
| radio | zip |
| photo | bmp |
| examen | txt |
| telephone | |
| famille | |
| tp | |

Il pourrait exister plusieurs fichiers portant le même nom mais ayant des extensions différentes, par exemple `controle.doc` et `controle.zip`.

Soient les variables suivantes :

```
const int MAX = 50;
string nomListeFichier, nomListeExtension;

string listeFichier[MAX];
int nombreFichiers = 0;

string listeExtension[MAX];
int nombreExts =0;
```

et en déclarant d'autres variables si nécessaire, écrire un programme pour vérifier l'existence des fichiers et en compte leur nombre. Le programme respecte les directives suivantes :

```
// l'usager donne le nom du fichier contenant la liste des fichiers

// l'usager donne le nom du fichier contenant la liste des extensions

// ouverture et existence des fichiers

// à partir du fichier ayant la liste des noms de fichiers
// stocker  dans le tableau listeFichier et conserver
// le nombre d'éléments

// à partir du fichier ayant la liste des noms des extensions
// stocker  dans le tableau listeExtension et conserver
// le nombre d'éléments

// à partir des tableaux listeFichier et listeExtension
// compter le nombre de fichiers  qui existent

// afficher le nombre de fichiers qui existent
```

# Corrigé

## Question 1

```
int main()
{   const int MAX = 100;
     string  nomFichier;
     fstream  fichierLu;
     string phrase[MAX];
     bool estOuvert = false;
     bool continuerLecture = true;
     int nombreLignes ;
     while (!estOuvert)
     {
          cout << "  veuillez donner le nom du fichier ";
          cin  >> nomFichier;
          fichierLu.open(nomFichier, ios::in);
          if (!fichierLu.fail())
               estOuvert = true;
     }

      nombreLignes = 0;
      while(continuerLecture)
      {
            getline(fichierLu,phrase[nombreLignes]);
             if (fichierLu.eof())
                    continuerLecture = false;
             else
                   nombreLignes++;
      }
      // la fonction FormaterPourConsole permet d'afficher les
accents
      // dans la fenêtre console.
       for ( int i = 0 ; i < nombreLignes ; i++)
             cout << FormaterPourConsole(phrase[i]) <<endl;

      fichierLu.close();
      return 0;

}
```

## Question 2

```
int main()
 {// 1.déclarations des variables
// motSecret variables lue dans le fichier
// mot variable de l'usager
       string  motSecret, mot;
       char lettre;
       int nombreEssai;
// 2. lecture du mot secret dansun fichier
       ifstream  fichier;
       fichier.open("mot.txt");
       fichier >>motSecret;
//3. mettre tous les lettres en minuscules du mot secret
        for ( int i = 0; i < motSecret.size(); i++)
             motSecret[i] = tolower(motSecret[i]);
// 4. mettre '_ dans les lettres de la variable mot
       bool continuer = true;
        for  ( int i = 0; i < motSecret.size(); i++)
              mot +='_';
// 5. boucle  tant que l'on n'a pas trouvé le mot
// dans la boucle  afficher les lettres trouvées
// et les caractères _ pour les lettres manquantes
// déterminer le nombre d'essai
// demander à l'usager une lettre et trouver si elle
//  est dans le mot secret, si oui, placer la lettre
// dans le mot
     nombreEssai =0;
       while (continuer)
       {
            for ( int i = 0 ; i < mot.size();i++)
                    cout << mot[i]<<" ";
              cout << endl;

             cout << "donner une lettre ";
             cin >> lettre;
             nombreEssai++;
             for (int i =0; i < motSecret.size(); i++)
                   if (motSecret[i] == lettre)
                         mot[i] = lettre;

              if (mot == motSecret)
                   continuer = false;
       }
// afficher le nombre d'essais
        cout << " bravo vous avez trouve le mot en " << nombreEssai<<
" essai "<<endl;

       return 0;
 }
```

## Question 3

```
int main()
 {
        string  lettre ="abcdefghijklmnopqrstuvwxyz";
        string phraseLisible;
        string phraseCryptee;
        int decalage, position;

         // décrypter
        bool trouver = false;
        decalage = 0;
        string unePhrase;
        while (!trouver )
        {
            for ( int i = 0; i < phraseCryptee.size(); i++)
            {   if (isalpha(phraseCryptee[i]))
                  {
                      position = int(phraseCryptee[i]-'a') ;
                       position= (26 +(position- decalage)) %26;
                       unePhrase +=lettre[position];
                  }
            else
                  unePhrase += phraseCryptee[i];
         }
          if (phraseLisible == unePhrase)
                  trouver = true;
            else
            {
                  unePhrase ="";
                   decalage++;
            }
        }

          cout << "le décalage est " << decalage <<endl;

            return 0;
 }
```

## Question 4

```
int main()
{
      const int MAX = 50;
      string nomListeFichier, nomListeExtension, nomTemp;
      ifstream fichierListe, fichierExt, fichierTemp;
      bool continuer;
      string listeFichier[MAX];
      int nombreFichiers = 0;

      string listeExtension[MAX];
      int nombreExts =0;

      int nombreFichiersExistants  = 0;

      // l'usager donne le nom du fichier contenant la liste des
fichiers
      cout << " veuillez donner le nom du fichier de la liste des
fichiers ";
      cin >> nomListeFichier;
      // l'usager donne le nom du fichier contenant la liste des
extensions
      cout << " veuillez donner le nom du fichier de la liste des
extensions ";
      cin >> nomListeExtension;

      // ouverture des fichiers
      fichierListe.open(nomListeFichier);
      fichierExt.open(nomListeExtension);
      // vérifier l'existence des fichiers
       if (!fichierListe.fail() && ! (fichierExt.fail()))
       {  continuer = true;
          // à partir du fichier ayant la liste des noms de fichiers
         // stocker  dans le tableau listeFichier et conserver
       // le nombre d'éléments
         while (continuer)
            {
                  fichierListe >>listeFichier[nombreFichiers];
                  if (fichierListe.eof())
                        continuer = false;
                  else
                        nombreFichiers++;
            }
            fichierListe.close();

            // à partir du fichier ayant la liste des noms des
extensions
        // stocker  dans le tableau listeExtension et conserver
      // le nombre d'éléments
            continuer = true;

            while (continuer)
            {
                  fichierExt >> listeExtension[nombreExts];
                  if (fichierExt.eof())
                        continuer = false;
```

```
                else
                      nombreExts++;
            }
            fichierExt.close();
             // à partir des tableaux listeFichier et listeExtension
              // compter le nombre de fichiers  qui existent
            for ( int i = 0 ; i < nombreFichiers; i++)
                      for  (int j = 0 ; j < nombreExts; j++)
                      {  nomTemp =
listeFichier[i]+"."+listeExtension[j];
                        fichierTemp.open(nomTemp);
                        if (!fichierTemp.fail())
                        {    nombreFichiersExistants++;
                            fichierTemp.close();
                        }
                      }
             // afficher le nombre de fichiers qui existent
           cout << " le nombre de fichier existants est "
                 << nombreFichiersExistants<<endl;

       }
       else
             cout << " fichiers inexistants " <<endl;

     return 0;
}
```