ÉCOLE POLYTECHNIQUE MONTRÉAL

Le génie sans frontières

Questionnaire examen final

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**	**20083**
Professeur	***Local***	***Téléphone***
Martine Bellaïche, responsable + chargés de cours	**A-201**	**4709 - 4313**

Jour	*Date*	*Durée*	*Heures*
Samedi	**13 décembre 2008**	**3h30**	**13h30 – 16h00**

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

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.
- On ne répond à aucune question. En cas de doute, veuillez faire vos suppositions et les écrire sur le cahier d'examen.

Bonne chance à tous!

Important

Cet examen contient **4** questions sur un total de **6** pages **(excluant cette page)**

La pondération de cet examen est de **40** **%**

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.

1. Questions générales (5 points)

1.1. Pour les questions suivantes, donnez l'identificateur de la fonction, le type de retour, le nombre, le type et l'identificateur des paramètres.

a) Écrivez seulement la déclaration d'une fonction qui retourne un nombre aléatoire entre les valeurs `inf` et `sup`.

b) Écrivez seulement la déclaration d'une fonction qui calcule, dans un tableau d'entiers de 10 lignes par 20 colonnes, le nombre d'éléments qui sont plus petits, qui sont égaux et qui sont plus grands à un nombre entier.

c) Écrivez seulement la déclaration d'une fonction qui vérifie si 3 valeurs entières sont en ordre croissant. Si les 3 valeurs sont en ordre croissant, la fonction retourne la valeur `true`, sinon elle retourne la valeur `false`.

1.2. Soient les déclarations suivantes :

```
int tableau[10] ={1, 3, -2, 10, 4, 9, -5, 15, 6, 5};
int i, somme = 0;
```

Pour les boucles de répétition suivantes, trouvez l'initialisation, l'expression booléenne et l'actualisation.

```
for (initialisation; expression booléenne; actualisation)
     cout << tableau[i]<<endl;
```

a) Affichez tous les éléments du tableau.

b) Affichez le contenu des éléments du tableau aux indices pairs du tableau.

c) Affichez le contenu des éléments du tableau tant que leur somme ne dépasse pas la valeur 15 et le nombre d'éléments du tableau.

1.3 Soit la structure suivante :

```
struct typeJoueur{
      string nom;
      int numero;
}
```

a) Écrivez les instructions qui déclarent un pointeur sur une structure `typeJoueur` et créent un espace mémoire qui sera initialisé à `{"Cristiano Ronaldo", 10}`.

b) Écrivez les instructions qui permettent de créer un tableau dynamique représentant une équipe de 7 joueurs.

c) Écrivez les instructions qui affectent au 3^{e} joueur du tableau dynamique le joueur créé en a).

1.4 Supposons que l'on nous donne un fichier binaire "ENTIERS.DAT" ne contenant que des entiers. Écrivez les déclarations et les instructions pour ouvrir le fichier et affecter à zéro l'avant-dernier entier sans réécrire le reste du fichier.

1.5 Soient les nombres hexadécimaux A = $(E6)_{16}$ et B = $(89)_{16}$, représentant des nombres entiers en complément à 2 sur 8 bits. Donnez le résultat de A-B en hexadécimal.

Solution

1.a
```
int aleatoire(int inf, int sup);
```
1.b
```
void statistique(int tableau[10][20], int nombre, int & nombreInf,
                 int & nombreSup, int & nombreEgal);
```
1.c
```
bool verifie(int n1, int n2, int  n3);
```

2.

for (i = 0; i < 10; i++)
for (i = 0; i < 10; i = i+2)
for (i = 0; i < 10 & (somme= somme+tableau[i]) <15; i++)

3
```
struct type_joueur{
string nom;
int numero;};

type_joueur *ptrJoueur;
ptrJoueur = new type_joueur;

ptrJoueur->nom = "Christiano Ronaldo";
ptrJoueur->numero = 10;

type_joueur *tabJoueur;
tabJoueur = new type_joueur[7];
tabJoueur[2] = *ptrJoueur;
```
4

```
int zero = 0;

fstream fichier;
fichier.open("ENTIERS.DAT", ios::binary|ios::in|ios::out);
if(!fichier.fail()){
      fichier.seekp(-1*sizeof(int),ios::end);
      fichier.write((char*)&zero, sizeof(int));
      fichier.close();
}
```

5
5D

2. Fonctions (4 points)

Écrivez une fonction appelée `remplacer()` qui permet de remplacer toutes les occurrences d'un caractère par le caractère suivant dans l'alphabet, (remplacez 'z' par 'a ' et 'Z' par 'A'). Qu'il soit en majuscule ou en minuscule, le caractère sera remplacé.

La fonction utilise trois paramètres : la chaîne de caractères (`string`), le caractère à remplacer et le nombre de remplacements effectués. La fonction retournera `false` si aucun remplacement n'a été fait et `true` sinon.

Par exemple, si nous appelons la fonction en lui transmettant la chaîne de caractères « `adieu veaux, vaches et zebres` » :

- si le caractère à remplacer est `'v'` la chaîne de caractères deviendra : « `adieu weaux waches et zebres` ». La fonction indiquera qu'il y a eu 2 remplacements.

- si le caractère à remplacer est `'z'` la chaîne de caractères deviendra : « `adieu veaux vaches et aebres` ». La fonction indiquera qu'il y a eu 1 remplacement.

Solution :

```
#include <iostream>
#include <string>
using namespace std;
bool remplacer(string &ch, char c, int &nbRemplacements)
{
      nbRemplacements = 0;
      char cr;
      if (c == 'z')
          cr = 'a';
      else if (c == 'Z')
          cr = 'A';
      else
          cr = char(c+1);

      for (int i = 0; i < ch.size(); i++)
      {
            if (ch[i] == c)
            {  ch[i]= cr ;
               nbRemplacements++;
            }
      }
      if (nbRemplacements == 0)
            return false;
      else
            return true;

}
```

3. Fichier binaire (5 Points)

ID3 est le nom des métadonnées qui peuvent être utilisées pour stocker des informations dans un fichier audio, comme par exemple les MP3. Les informations que ID3 (dans sa première version) permettent d'insérer sont les suivantes :

```
struct ID3{

  char TAG[3];
  char titre[30];
  char artiste[30];
  char album[30];
  char annee[4];
  char commentaire[30];
  char genre;
};
```

3.1 Écrivez une fonction **lireDonnees** qui lit un fichier binaire contenant une séquence d'enregistrements de type ID3. En particulier, la fonction a comme paramètres : le nom d'un fichier binaire, un tableau d'enregistrements de type ID3 et un entier qui représente le nombre d'enregistrements lus. La fonction doit lire le fichier et remplir correctement les informations du tableau et le nombre d'enregistrements. Elle retourne un booléen : `true` si la lecture est bien réussie, sinon `false`.

3.2 Écrivez une fonction **changeArtiste** qui doit trouver un enregistrement correspondant à un titre, changer le nom d'un artiste, et enfin sauvegarder ce changement sur le fichier binaire original. La fonction reçoit le tableau d'enregistrements de type ID3, le nombre d'éléments contenus dans le tableau, le titre de la chanson (on suppose qu'il n'y a pas de doublon), le nouveau nom d'artiste et enfin le nom du fichier original. La fonction doit mettre à jour le fichier sans réécrire les enregistrements inchangés. La fonction retourne un booléen : `true` si la chanson a été trouvée et correctement sauvegardée, sinon `false`. Les comparaisons et les affectations peuvent se faire sur le type `string` ou sur un tableau de caractères.

Indice: Si on a les déclarations suivantes:
`char chaine1[30], chaine2[30];`

La fonction `strcmp(chaine1,chaine2)` retourne la valeur 0 si `chaine1` est égal à `chaine2`.

La fonction `strcpy(chaine1, chaine2)` copie le contenu de `chaine2` dans `chaine1`.

Solution

```
bool lireDonnees(string nomFichier, ID3 tableau[], int &nbID3){
	ifstream fichier;

	fichier.open(nomFichier.c_str(), ios::binary);
	if(fichier.fail()){
		return false;
	}
	else{
		fichier.seekg(0, ios::end);
		nbID3 = fichier.tellg()/sizeof(ID3);
		fichier.seekg(0, ios::beg);

		for(int i=0; i<nbID3; i++){
			fichier.read((char*) &tableau[i], sizeof(ID3));
		}
		fichier.close();
		return true;
	}
}

bool changeArtiste(ID3 tableau[], int nbID3, string titreChanson, string artiste, string
nomFichier){
	int k = -1;
	bool trouve = false;
	for(int i=0; i<nbID3 && !trouver; i++){
		if(strcmp(tableau[i].titre, titreChanson.c_str())==0){
			k = i;
			trouver = true;
		}
	}
	if(trouver){
		fstream fichier;
		fichier.open(nomFichier.c_str(), ios::binary|ios::in|ios::out);
		if(fichier.fail()){
			return false;
		}
		else{
			strcpy(tableau[k].artiste, artiste.c_str());
			fichier.seekp(k*sizeof(ID3),ios::beg);
			fichier.write((char*)&tableau[k], sizeof(ID3));
			fichier.close();
			return true;
		}
	}
	else{
		return false;
	}
}
```

4. Allocation dynamique (6 points)

On désire conserver dans une variable de `typeToutGenre`, un répertoire de chansons.

Soient les structures suivantes

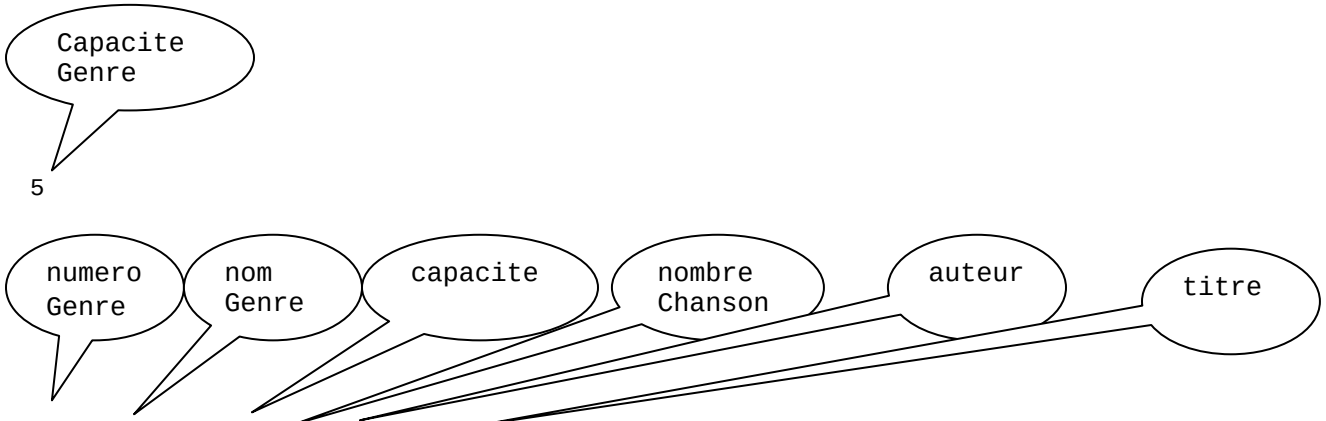
```
struct typeChanson
{  string auteur;
   string titre;
};
struct typeGenre
{  int  numeroGenre;
   string nomGenre;
   int capacite;
   int nombreChanson;
   typeChanson* listeChanson;
};
struct typeToutGenre
{
   int capaciteGenre;
   typeGenre** ensChanson;
};
```

La structure `typeChanson` permet de stocker le titre et l'auteur d'une chanson. La structure `typeGenre` permet pour un genre donné, de conserver la liste des chansons dans le champ `listeChanson` représentant un tableau dynamique dont la taille est stockée dans le champ `capacite`, et le nombre de chansons contenues dans le tableau est stocké dans le champ `nombreChanson`. La structure `typeToutGenre` conserve tous les genres de chansons; le champ `ensChanson` est un tableau dynamique dont chaque élément est un pointeur vers une structure `typeGenre`; le champ `capaciteGenre` représente la taille de ce tableau dynamique.

Écrivez la fonction `lectureFichier` qui lit les informations des chansons d'un fichier texte et les stocke dans le paramètre `monRepertoire`.

```
bool lectureFichier(string nomFichier, typeToutGenre& monRepertoire)
```

Le fichier texte a le format suivant :



4. Allocation dynamique (suite)

Attention : Le nombre entier sur la première ligne du fichier ne correspond pas au nombre de lignes à lire par la suite.

Pour répondre à la question, veuillez suivre les commentaires.

```
bool lectureFichier(string nomFichier, typeToutGenre & monRepertoire)
{
      int numero;
      ifstream fichier;
      // ouverture du fichier
      // vérification de l'existence du  fichier
      // lecture de capaciteGenre
      // vérifier si le fichier n'est pas vide
      // allouer dynamiquement  dans ensChanson un tableau de type typeGenre*
      // initialiser  tous les éléments de ensChanson à 0
      // lire   un numero
      // tant que non fin de fichier
      //    allouer à la position numero-1 du tableau ensChanson un espace mémoire
      //    de typeGenre
      //    À l'élément numero-1 du tableau ensChanson initialiser le champ numeroGenre
      //    lire et initialiser  à l'élément numero-1 du tableau ensChanson, les champs
      //    nomGenre, capacite, nombreChanson
      //    Dans le champ listeChanson de l'élément ensChanson[numero -1],
      //    allouer un espace mémoire de type typeChanson et de taille capacite
      //    Faire une boucle sur  nombreChanson pour lire les auteurs et
      //    les titres des chansons et les stocker dans le tableau listeChanson
      //    lire un autre numéro
      // fermer le fichier
}
```

Solution

```
bool lectureFichier(string nomFichier, typeToutGenre & monRepertoire)
{
  int numero;
  ifstream fichier;
  fichier.open(nomFichier.c_str());
  if (!fichier.fail())
  {  fichier>>monRepertoire.capaciteGenre;
     if (!fichier.eof())
     {  monRepertoire.ensChanson = new typeGenre*[monRepertoire.capaciteGenre];
        if (monRepertoire.ensChanson== 0)
            return false;
        for ( int i = 0 ; i < monRepertoire.capaciteGenre;i++)
          monRepertoire.ensChanson[i] = 0;
        fichier >> numero;
        while(!fichier.eof())
        {
          monRepertoire.ensChanson[numero-1] = new typeGenre;
          if ( monRepertoire.ensChanson[numero-1] == 0)
             return false;
          monRepertoire.ensChanson[numero-1]->numeroGenre = numero;
          fichier>> monRepertoire.ensChanson[numero-1]->nomGenre;
          fichier>>monRepertoire.ensChanson[numero-1]->capacite;
          fichier>>monRepertoire.ensChanson[numero-1]->nombreChanson;
          monRepertoire.ensChanson[numero-1]->listeChanson=
           new typeChanson[monRepertoire.ensChanson[numero-1]->capacite];
          if (monRepertoire.ensChanson[numero-1]->listeChanson == 0)
             return false;
          for ( int i = 0; i <monRepertoire.ensChanson[numero-1]->nombreChanson; i++)
          { fichier>>monRepertoire.ensChanson[numero-1]->listeChanson[i].auteur;
            fichier >>monRepertoire.ensChanson[numero-1]->listeChanson[i].titre;
          }
          fichier>>numero;
        }
        fichier.close();
      }
      else
         return false;

      return true;

      }
      else
            return false;
}
```