Structures de données et algorithmes fondamentaux
Examen final
Durée : 120 minutes
Tous les documents sont autorisés

Nom et prénom : _____

Question :	1	2	3	4	5	6	7	Somme
Barème :	4	2	2	5	2	3	2	20
Points :								

1. (4 points) Donner les réponses correctes.

   (a) La taille d'un vecteur ne peut pas être changée après création.
   ◯ vrai √ **faux**

   (b) La déclaration "`vector<int, int> vec`" crée un vecteur de vecteurs d'entiers.
   ◯ vrai √ **faux**

   (c) L'accès `vec.at(i)` à l'élément à l'indice `i` du vecteur `vec` lance une exception si `i` est strictement négatif.
   √ **vrai** ◯ faux

   (d) Bien que ça ne soit pas conseillé, une donnée membre peut être publique.
   √ **vrai** ◯ faux

   (e) Toute classe doit avoir au moins une fonction membre.
   ◯ vrai √ **faux**

   (f) Sous l'hypothèse que les éléments d'un vecteur d'entiers sont ordonnés de manière *décroissante*, la recherche d'un entier donné peut être faite en temps $O(\log n)$ où n est la taille du vecteur.
   √ **vrai** ◯ faux

   (g) L'algorithme du tri par sélection prend temps $O(n^2)$ où n est la taille du vecteur à trier.
   √ **vrai** ◯ faux

   (h) Il n'est pas possible pour une fonction de s'appeler elle-même.
   ◯ vrai √ **faux**

2. (2 points) Écrire une fonction qui prend un vecteur de strings et qui renvoie un élément de longueur maximale. S'il y a plusieurs éléments avec la longueur maximale, vous pouvez choisir librement un de ces éléments. Vous pouvez supposer que le vecteur n'est pas vide.

Solution:

```
string longest_string(const vector<string> &vec)
{
        int maxlen = 0;
        int maxind = 0;

        for(int i = 0; i < vec.size(); ++i)
        {
                if(vec[i].length() > maxlen)
                {
                        maxlen = vec[i].length();
                        maxind = i;
                }
        }

        return vec[maxind];
}
```

3. (2 points) Écrire une fonction qui prend un vecteur d'entiers et qui supprime le dernier élément (s'il existe).

Solution:

```
void delete_last(vector<int> &vec)
{
        if(!vec.empty())
                vec.erase(vec.begin() + vec.size() - 1);
}
```

4. (5 points) Écrire une classe `Phrase` décrite comme suit. Elle doit contenir une suite de strings et des fonctions pour effectuer les opérations suivantes :

   — Rajouter un string à la fin de la suite de strings.
   — Vérifier si la phrase est une question, c'est-à-dire si le dernier string dans la suite finit avec le caractère '?'.
   — Calculer la longueur totale de la phrase, c'est-à-dire la somme des longueurs des strings dans la suite.

   Spécifier le fichier (`.h` ou `.cpp`) pour les différentes parties de code que vous écrivez. Il n'est pas nécessaire de donner les instructions `using`, `#include`, `#ifndef`, `#define` et `#endif`.

Solution: Dans le fichier `phrase.h` :

```
class Phrase {
private:
        vector<string> mots;

public:
        void rajouter(const string &str);
        bool question() const;
        int longueur() const;
};
```

Dans le fichier `phrase.cpp` :

```
void Phrase::rajouter(const string &str)
{
        mots.push_back(str);
}

bool Phrase::question() const
{
        if(mots.empty())
                return false;

        int last = mots.size() - 1;

        if(mots[last].empty())
                return false;

        return (mots[last][mots[last].length()-1] == '?');
}
```

Solution:

```
int Phrase::longueur() const
{
        int sum = 0;
        for(int i = 0; i < mots.size(); ++i)
                sum += mots[i].length();
        return sum;
}
```

5. (2 points) Écrire un programme qui laisse l'utilisateur saisir une suite d'entiers jusqu'à ce qu'il mette l'entier 42. Ensuite, le programme affiche tous les entiers saisis (42 inclus) dans l'ordre inverse, en omettant les entiers nuls.

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
```

Solution:

```
    vector<int> ints;
    int next;

    do {
            cin >> next;
            ints.push_back(next);
    } while(next != 42);

    for(int i = ints.size()-1; i >= 0; --i)
            if(ints[i] != 0)
                    cout << ints[i] << endl;
```

```
}
```

6. (3 points) Soit donnée la classe suivante :

```
class vec_trie {
private:
        vector<int> elems;

public:
        int rang(int x) const;
        void rajouter(int x);
};
```

Pour accélérer la recherche d'un élément, la fonction membre `rajouter` garantit que le vecteur `elems` reste toujours trié.

(a) Compléter l'implémentation de la fonction membre `rang` pour trouver le nombre d'éléments du vecteur qui sont inférieurs ou égaux à l'entier `x`. Faire attention à la complexité en temps de votre algorithme.

```
int vec_trie::rang(int x) const
{
```

Solution:

```
        int debut = 0;
        int fin = elems.size();

        while(debut < fin) {
                int mil = (debut + fin)/2;

                if(x < elems.at(mil))
                        fin = mil ;
                else // x >= elems.at(mil)
                        debut = mil + 1;
        }

        return debut;
```

Solution:

}

(b) Quelle est la complexité en temps de votre algorithme ? Pourquoi ?

Solution: $O(\log n)$ où $n = \texttt{elems.size()}$, car $\texttt{fin} - \texttt{debut}$ est au moins divisé par 2 dans chaque itération :
Dans le premier cas (`if`), nous avons

$$\texttt{fin}' - \texttt{debut}' = \left\lfloor \frac{\texttt{debut} + \texttt{fin}}{2} \right\rfloor - \texttt{debut} \leq \frac{\texttt{fin} - \texttt{debut}}{2} \ .$$

Dans le deuxième cas (`else`), nous avons

$$\texttt{fin}' - \texttt{debut}' = \texttt{fin} - \left\lfloor \frac{\texttt{debut} + \texttt{fin}}{2} \right\rfloor - 1 \leq \frac{\texttt{fin} - \texttt{debut}}{2} \ .$$

7. (2 points) Considérer les fonctions suivantes :

```
void f(int n)
{
        for(int i = 0; i < n; ++i)
        {
                cout << ".";
                ++i;
        }
}

void g(int n)
{
        for(int i = 0; i < n; ++i)
        {
                for(int j = 0; j < n-i; ++j)
                {
                        cout << ".";
                }
        }
}
```

(a) Quel est le nombre exact d'itérations de la boucle dans la fonction `f` ? Expliquer.

Solution: $\lfloor \frac{n}{2} \rfloor$ car la boucle est équivalente à celle-ci :

```
        for(int i = 0; i < n; i = i + 2)
        {
                cout << ".";
        }
```

(b) Quel est le nombre d'itérations de la boucle intérieure dans la fonction g exprimé en notation de Landau ? Expliquer.

Solution: $O(\mathtt{n}^2)$

Les valeurs de $\mathtt{n}-\mathtt{i}$, et donc le nombres d'itérations de la boucle intérieure, dans les itérations de la boucle extérieure sont

$$\mathtt{n},\ \mathtt{n}-1,\ \mathtt{n}-2,\ \ldots,\ 1\ .$$

Leur somme est égale à

$$\sum_{k=1}^{\mathtt{n}} k = \frac{\mathtt{n}\cdot(\mathtt{n}+1)}{2} = O(\mathtt{n}^2)\ .$$