**Contrôle Continu AP4A – Mardi 25 Avril 2023 - durée 1 heure
(aucun document autorisé)**

**I.** (2 pts) Quelle est la différence entre une copie de surface et une copie profonde entre deux objets. Dans quel cas utiliser l’une ou l’autre et qu’est-ce que cela implique dans l’écriture de la classe.

**II.** (4 pts) Considérons le type Vecteur en C++ tel que réalisé en TP et des variables de type Vecteur. Donner la notation fonctionnelle équivalente des instructions suivantes :

Vecteur u, v;

... // les vecteurs sont initialises avec des contenus

v[i] = v[j] = 2.0;

cout << “Vecteur u : “ << u << “Vecteur v : “ << v << endl;

if (u != v) u = v;

++u = v++;

**III**. (6 pts) On considère la classe Liste telle que codée en récursif, supposons que l’opérateur concat a été écrit comme ceci :

template <class T>

class Liste {

 T tete; // tete de liste = element

 Liste\* reste; // suivi d'une liste

public:

 Liste **concat**(Liste& l) {

 Liste\* l1 = new Liste(\*this);

 return l1->autoConcat(l);

 }

};

Donner un schéma d’occupation mémoire précis (pile et tas) pour le programme suivant aux points d’exécution 1), 2), 3) et 4), puis ensuite donner une version correcte de **concat**.

{

 Liste<float> l;

 l.ajouter(1.0, 0);// ajouter

 l.ajouter(2.0, 0);

 l.ajouter(3.0, 0);

 l.ajouter(4.0, 0);

 Liste<float>\* pl = new Liste<float>;

 pl->ajouter(1.0, 0);// ajouter

 pl->ajouter(2.0, 0);

 pl->ajouter(3.0, 0);

 pl->ajouter(4.0, 0);

 //**1)**

 Liste<float> m;

 m = l.concat(\*pl) ;

 //**2)**

 delete pl;

 //**3)**

}//**4)**

**IV.** (4 pts) Définissez une classe Vehicule avec des attributs très généraux comme :

* la marque
* la date d'achat
* le prix d'achat
* le prix courant

Définissez un constructeur prenant en paramètre les trois attributs correspondant à la marque, la date d'achat et le prix d'achat (le prix courant sera pour plus tard). Définissez une méthode publique qui affiche l'état de l'instance, ou bien surcharger l'operateur ostream& operator<<(ostream&, const Vehicule&).

**V.** (4 pts)Étant donné un graphe orienté, on veut implémenter une structure de données de graphe en C++. On utilisera la liste d'adjacence où chaque sommet du graphe stocke une liste de sommets adjacents. Les sommets sont identifiés à des entiers de 0 à n-1. Compléter le programme ci-dessous au niveau des « **...** » :

#include <iostream>

#include “vector.h”

#include “list.h”

using namespace std;

// Data structure to store a graph edge

struct Edge {

 int src, dest;

};

// A class to represent a graph object

class Graph

{

public:

 // a vector of lists to represent an adjacency list

 vector<list<int>> adjList;

 // Graph Constructor

 Graph(vector<Edge> const &edges, int n)

 {

 ...

 }

 // Print adjacency list representation of a graph

 void printGraph(Graph const &graph, int n)

 {

 ...

 }

Extrait de https://www.techiedelight.com/fr/graph-implementation-using-stl/