Rattrapage

Exercice 1 Ecrire trois fonctions `v(int x)`, `r(int x)`, `p(int x)` utilisant respectivement le passage par valeur, par référence et par pointeur, permettant chacune d'incrémenter la *left-value* `x` fournie en argument. Vous donnerez dans une même fonction `main()` un exemple d'utilisation pour chaque fonction.

Exercice 2 Créer une classe `point` simulant un point du plan dont les coordonnées de type `double` seront des attributs privés et munissez la de **exactement deux méthodes** : un constructeur et une fonction membre `proche` renvoyant `true` si la distance du point appelant à celle d'un point en paramètre est inférieure à 1 (sans utiliser `include`). Faire, sans toucher à la classe `point`, une fonction binaire **non-membre** `proche2` avec la même fonctionalité.

Exercice 3 Qu'affiche précisément ce programme ?

```
#include <iostream>
using namespace std;
main()
{ int t[4]={20,30,40,50};
  int * ad[4];
  for(int i=0; i<4; i++) ad[i]=t+i;
  for(int i=0; i<4; i++) cout << * ad[i] << " "; cout << "\n";
  cout << * (ad[1]+2) << " " << * ad[1] + 1 << "\n";
}
```

Exercice 4 Créer une classe `vect` simulant un vecteur de l'espace euclidien de dimension variable. On mettra en attribut privé `int dim` la dimension du vecteur et `double * adr` l'adresse du vecteur (dont les composantes seront de type `double`). Faire les méthodes suivantes pour gérer la partie dynamique et libérer l'espace mémoire : un constructeur prenant la dimension en paramètre, un destructeur, un constructeur de recopie et l'opérateur = surdéfini.

Exercice 5 Faire une fonction `pascal(int n)` affichant les n premières lignes du triangle de Pascal. Par exemple l'appel `pascal(7)` affiche :

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1