Examen

La propreté de la programmation est cruciale pour l'obtention de tous les points du barême.

Exercice 1 (3 points) Donnez le code de la fonction `facto` transformant un nombre n en son factoriel $n! = n \times (n-1) \times \ldots \times 2$, par exemple on déclare `int x=4;` puis après l'appel on a $x = 24$, dans les cas suivants:
1) L'appel avait été effectué par l'instruction `facto(&x);`
2) L'appel avait été effectué par l'instruction `facto(x);`
3) L'appel avait été effectué par l'instruction `x=facto(x);`

Exercice 2 1) (5 points) Ecrire une classe cannonique `Point` simulant les points de $\mathbb{R}^n$ (espace géométrique usuel de dimension n, c'est-à-dire que le nombre de coordonnées est égale à n) avec: deux attributs privés `int dim` et `double* coord`, un constructeur, un destructeur, un constructeur de recopie, et l'opérateur `=`.

2) (2 points) Munir la classe `Point` des opérateurs `==` et `-` en tant que fonctions amies.

Exercice 3 (4 points) Faire une fonction `int** arrangement(int m)` construisant la table des valeurs de

$$ar(n,p) := \binom{n}{p} \times p! = n \times \ldots \times (n-p+1)$$

pour $n = 0,1,\ldots,m$ et $0 \leq p \leq n$. Par exemple l'appel `arrangement(4)` construit un tableau :

1
1 1
1 2 2
1 3 6 6
1 4 12 24 24

On procédera **impérativement** de la façon décrite ci-dessous:
1) On utilisera un tableau de tableaux via un pointeur de pointeurs `int** mat`. Le tableau de tableaux contient $m+1$ tableaux `mat[i]` de taille $i+1$ pour $i = 0,1,\ldots,m$. On fera l'allocation de mémoire avec $m+2$ appels à `new` et la désallocation avec $m+2$ appels à `delete[]`. Par exemple `int** mat = new int* [m+1];` pour le tableau de tableaux...
2) On initialisera `mat` de la façon suivante : `mat[i][0]=1` pour tout i. Puis on complètera `mat` avec la relation suivante $ar(n,p) = p \times ar(n-1,p-1) + ar(n-1,p)$ avec $ar(n-1,p) = 0$ pour $p = n$. Ainsi la valeur de `mat[i][j]` sera égale à $ar(i,j)$.

Problème 1 (6 points)

Une fonction norme de $\mathbb{R}^n$ est une fonction qui associe un réel positif à chaque point $x = (x_1,x_2,\ldots,x_n) \in \mathbb{R}^n$ et satisfait des axiomes. La norme usuelle est la norme euclidienne, ou norme 2 qui est $N_2(x) = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$. Mais il existe d'autres normes comme: La norme 1 est $N_1(x) = |x_1| + |x_2| + \ldots + |x_n|$ avec $|x_i| = x_i$ si $x_i \geq 0$ et $-x_i$ sinon. La norme inf est $N_\infty(x) = \max\{|x_1|,|x_2|,\ldots,|x_n|\}$.

A chaque fonction norme $N(x)$ on peut associer une fonction distance $d(x,y) = N(x-y)$. On a ainsi la distance 1, la distance 2 (distance euclidienne) et la distance inf.

A chaque distance $d(x,y)$ on associe le concept de boule $B(x,r) = \{y : d(x,y) \leq r\}$, un point $y \in \mathbb{R}^n$ appartient donc la boule de centre $x \in \mathbb{R}^n$ et de rayon $r \in \mathbb{R}$ si sa distance à x est inférieure au rayon. Ainsi on a la boule 1, la boule 2 et la boule inf.

En supposant que vous disposez des sources de la classe `Point` (Exercice 2), vous implémenterez une classe mère `Boule` ayant une fonction virtuelle `distance` que vous implémenterez dans trois classes dérivées `Boule1`, `Boule2` et `BouleINF`. Vous munirez également `Boule` d'une fonction `contient` testant si y est dans la boule, d'une fonction `gonfle` qui augmente le rayon de la boule d'un valeur donnée en paramètre, d'une fonction `intersecte` qui teste si deux boules de même nature intersectent ou non.

On utilisera pour cela le fait que $B(x,r) \cap B(x',r') \neq \emptyset$ si et seulement si $x' \in B(x,r+r')$.