5,25/20 : Les premières questions (Q1 à Q5) n'ont pas été rendues !

```
**********************/
#pragma once
#define _USE_MATH_DEFINES

#include <math.h>
#include "Point2D.h"
#include "Figure.h"

#define PI 3.14159265358979323846

class Point2D;
class Figure;

class Cercle : public Figure
{
private :
        double rayon;
        Point2D centre;

public:
        Cercle(double leRayon, Point2D leCentre);
        double getPerimetre();
        double getSurface();
};



/**********************
Commande.h
**********************/
#pragma once
#define _USE_MATH_DEFINES
#include <vector>
#include <string>
#include "Figure.h"

using namespace std;

class Figure;

class Commande
{
private:
        bool commandeTerminee;
        double prixMetreDecoupe , prixMetreCarreMatiere ;
        string idCommande;
        vector<Figure*> lesFigures;

public:
        Commande(string identifiantCommande , double lePrixMetreDecoupe , double lePrixMetreCarreMatiere);
        string getIdCommande() { return idCommande; }
        void ajouterNouvelleFigure(Figure *laFigure);
        void cloturerCommande();

        double getPrix() ;
};



/**********************
Figure.h
**********************/
#pragma once
#define _USE_MATH_DEFINES

class Figure
{
public:
        void getPerimetre();
        void getSurface();
};

/**********************
Point2D.h
**********************/
// Cette classe n'est pas à modifier
#pragma once

class Point2D
{
private:
        double x , y ;

public:
```

1 point

Ce sont des méthodes virtuelles (virtual) pures(=0)

```
        Point2D(double x=0 , double y=0);
        double getX();
        double getY();
        void setX(double newX);
        void setY(double newY);
};



/**********************
Polygone.h
**********************/
#pragma once
#define _USE_MATH_DEFINES
#include <vector>
#include "Figure.h"
#include "Point2D.h"

using namespace std;

class Point2D;
class Figure;

#define abs(x) ( (x) >=0 ? (x) : -(x) )

class Polygone : public Figure
{
protected:
        vector<Point2D *> lesSommets;
        bool estFerme;

public:
        Polygone(void);
        static double distance(const Point2D *p1, const Point2D *p2);
        void insereUnNouveauSommet(Point2D *leSommet, int position = -1);
        void fermeLePolygone();
        double getPerimetre();
        double getSurface();

};



/**********************
Cercle.cpp
**********************/
#include "Cercle.h"
#define _USE_MATH_DEFINES

Cercle::Cercle(double leRayon, Point2D leCentre)
{
        this->rayon = leRayon;
        this->centre = leCentre;
}

double Cercle::getPerimetre()
{
        return  2 * PI * rayon;
}

double Cercle::getSurface()
{
        return PI * rayon * rayon;
}


/**********************
Commande.cpp
**********************/
#include "Commande.h"
#define _USE_MATH_DEFINES

Commande::Commande(string identifiantCommande , double lePrixMetreDecoupe , double lePrixMetreCarreMatiere)
{
        this->idCommande = identifiantCommande;
        this->prixMetreDecoupe = lePrixMetreDecoupe;
        this->prixMetreCarreMatiere = lePrixMetreCarreMatiere;
}

void Commande::ajouterNouvelleFigure(Figure *laFigure)
{
        lesFigures.push_back(laFigure);
}

void Commande::cloturerCommande()
{
```

1 point

1 point

1,5 points

```
        this->commandeTerminee = true;
}

double Commande::getPrix()
{
        ............
}



/**********************
main.cpp
**********************/
#define _USE_MATH_DEFINES
#include <iostream>
#include <conio.h>
#include "Polygone.h"
#include "Cercle.h"
#include "Commande.h"

using namespace std ;                           // espace de nommage standard

int main()
{
        // Testez la classe Cercle
        Point2D cercle((0,0), 4 );



        // Testez la classe Polygone avec la figure de test du sujet
        double Coordonnees[6][2]={ { 1 , 1 } , { 3 , 5 } , { 5 , 7 } , { 5 , 1 } , { 3 , 3 } , { 3 , 1 } };


        ...



        // Sapin de Noel et boules
        double CoordonneesSapin[15][2]={ { 2 , 2 } , { 5 , 4 } , { 3 , 4} , { 5 , 6 } , { 4 , 6 } , { 6 , 8 },
{ 8 , 6 } , { 7 , 6 },
                                                                { 9 , 4 } , { 7 , 4} , { 10 , 2 } , { 6
.5 , 2 } , { 6.5 , 1 },  { 5.5 , 1 } , { 5.5 , 2 }};


        double CoordonneesCentreCercles[6][2]={ { 2.5 , 3.5 } , { 3.5  , 5.5 } , { 4.5 , 7.5 } , { 7.5  , 7.5 }
, { 8.5 , 5.5 } , { 9.5  , 3.5 } };
        int i;


        // Création du polygone sapin
        ...

        cout <<"superficie du sapin = " << .... << "   " ;
        cout <<"Perimetre du sapin = " << ... << endl;

        // Création des 6 cercles
        ...
                cout <<"superficie du cercle " << i <<" = " <<... << "   " ;
                cout <<"Perimetre du cercle  " << i <<" = " << ...<< endl;
        ...

        // Création de la commande du Père Noel
        ...

        // Ajout des figures (le sapin et les 6 cercles) à la commande
        ...

        // Affichage du prix de cette commande
        cout <<"\nCout de la commande : " << ... <<" = " << ... <<" euros" << endl;




        _getch();       // on attend l'appui sur une touche
        return 0 ;      // fin du programme
}



/**********************
Point2D.cpp
**********************/
// Cette classe n'est pas à modifier
#include "Point2D.h"
```

Test de la classe Cercle : 0,25 point

```
#define _USE_MATH_DEFINES

Point2D::Point2D(double x , double y)
{
        this->x = x;
        this->y = y;
}

double Point2D:: getX()
{ return x ;}

double Point2D::getY()
{
        return y;
}

void Point2D::setX(double newX)
{
        x = newX;
}

void Point2D::setY(double newY)
{
        y = newY;
}


/**********************
Polygone.cpp
**********************/
#define _USE_MATH_DEFINES
#include <math.h>
#include "Polygone.h"


Polygone::Polygone(void)
{
        this->estFerme = false;
}

double Polygone::distance(const Point2D *p1, const Point2D *p2)
{

}

void Polygone::insereUnNouveauSommet(Point2D *leSommet, int position)
{
        lesSommets.push_back(leSommet);
}

void Polygone::fermeLePolygone()
{
        this->estFerme = true;
}

double Polygone::getPerimetre()
{
        // addition de tout les côtés par le théorème de pythagore
}

double Polygone::getSurface()
{
        double surface = 0;
        int i, n = lesSommets.size();
        for (unsigned i = 0; i < n - 1; i++)
        {
                // formule de calcul de l'aire d'un polygone

                return surface;
        }
}
```

0,5 point