```
/***********************
Capteur.h
***********************/
```
5/5

```
#pragma once
using namespace std;
#include <string>
class Capteur
{
protected:
        string type, noSerie;
        static int nombreDeCapteurs;
public:
        Capteur(string type, string noSerie);
        string getType();
        string getNoSerie();
        static int getNombreCapteurs();
        ~Capteur();
};
```

```
/***********************
CapteurTemperature.h
***********************/
```
2/2

```
#pragma once
#include "Capteur.h"
class CapteurTemperature : public Capteur
{
protected:
        double temperatureMinAutorisee;
        double temperatureMaxAutorisee;

public:
        CapteurTemperature(string numeroSerie, double min, double max);
        double getTemperature;

};
```

```
/***********************
CapteurTemperatureExterieure.h
***********************/
```
2/2

```
#pragma once
#include "CapteurTemperature.h"
class CapteurTemperatureExterieure : public CapteurTemperature
{
public:
        CapteurTemperatureExterieure(string derniersChiffresNumeroSerie, double min, double max);
        bool verifieConformite;
};
```

```
/***********************
Capteur.cpp
***********************/
```
3/4

```
#include "Capteur.h"
#include <string>
int nombrecapteur = 0;
Capteur::Capteur(string type, string noSerie)
{
        this->type = type;
        this->noSerie = noSerie;
}

string Capteur::getType()
{
        return type;
}

string Capteur::getNoSerie()
{
        return noSerie;
}

int Capteur::getNombreCapteurs()
{
        return nombrecapteur;
}

Capteur::~Capteur()
{
        nombrecapteur--;
}
```

```
/**********************
CapteurTemperature.cpp
**********************/
#include "CapteurTemperature.h"
#include <string>

CapteurTemperature::CapteurTemperature(string numeroSerie, double min, double max) : Capteur(type, noSerie)
{

}
CapteurTemperature::getTemperature()
{
        return 0;
}

CapteurTemperature::temperatureMaxAutorisee()
{
        return max;
}


CapteurTemperature::temperatureMaxAutorisee()
{
        return mix;
}

/**********************
CapteurTemperatureExterieure.cpp
**********************/
#include "CapteurTemperatureExterieure.h"
#include <string>

CapteurTemperatureExterieure::CapteurTemperatureExterieure(string derniersChiffresNumeroSerie, double min, doubl
e max) :CapteurTemperature(temperatureMinAutorisee, temperatureMaxAutorisee)
{

}
CapteurTemperatureExterieure::verifieConformite()
{

}

/**********************
TestCapteurs_A_Completer.cpp
**********************/
/*************************************************
Programme principal à compléter
Au final, l'exécution du programme doit
produire un résultat similaire à l'exécutable fourni
*************************************************/
#define _CRT_SECURE_NO_WARNINGS

#include <iostream> // bibliothèque de gestion des E/S
#include <conio.h>      // gestion de la console (ici _getch())
#include <windows.h>
#include <vector>

#include "Capteur.h"
#include "CapteurTemperature.h"
#include "CapteurTemperatureExterieure.h"



using namespace std; // utilisation de l'espace de nommage standard


/*================================================
Fonction principale
==========================================*/
int main()
{
        // Prise en compte des accents
        // Il faudra choisir la police de caractères Consolas ou Lucida
        SetConsoleOutputCP(1252);

        //==== Déclaration d'éventuelles variables locales ====
        //...
        Capteur* TEMP[20];
        float t;
        // Affichage du nombre de capteurs
        cout << "ETAPE1 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 0)" << endl << endl;

        TEMP[0] = new Capteur("111101");
        TEMP[0] -> t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[1] = new Capteur("111102");
```

0/7

0/3

2/17

1 pt

1 pt

```
        TEMP[1] -> t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[2] = new Capteur("111103");
        TEMP[2]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[3] = new Capteur("111104");
        TEMP[3]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[4] = new Capteur("111105");
        TEMP[4]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[5] = new Capteur("111106");
        TEMP[5]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[6] = new Capteur("111107");
        TEMP[6]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[7] = new Capteur("111108");
        TEMP[7]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[8] = new Capteur("111109");
        TEMP[8]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[9] = new Capteur("111110");
        TEMP[9]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[10] = new Capteur("111111");
        TEMP[10]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[11] = new Capteur("111112");
        TEMP[11]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[12] = new Capteur("111113");
        TEMP[12]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[13] = new Capteur("111114");
        TEMP[13]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[14] = new Capteur("111115");
        TEMP[14]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[15] = new Capteur("111116");
        TEMP[15]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[16] = new Capteur("111117");
        TEMP[16]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[17] = new Capteur("111118");
        TEMP[17]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[18] = new Capteur("111119");
        TEMP[18]->t = rand() <= 80.5 && t = rand() >= -20.5;
        TEMP[19] = new Capteur("111120");
        TEMP[19]->t = rand() <= 80.5 && t = rand() >= -20.5;


        // Créer un tableau de 20 capteurs de température (modèles de base)
        // Ces  capteurs de temperature  ont des numéros de séries allant de "111101" à "111120"
        // de temperature mini -20.5 et temperature maxi +80.5
        // ...

        // Affichage du nombre de capteurs
        cout << "ETAPE2 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 20)" << endl << endl;

        // Afficher 8 mesures par capteurs (20 lignes de 8 valeurs)
        cout << "CAPTEURS DE TEMPERATURE (MODELE DE BASE) " << endl;
        cout.precision(4);      // 4 digits de précision
        //...

        // Supprimer de la mémoire 10 capteurs
        //...

        // Affichage du nombre de capteurs
        cout << "ETAPE3 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 10)" << endl << endl;

        // Créer 12 capteurs de température extérieure dont les numéros de séries finissent par "5501" à "5512"
        // Les 11 premiers capteurs ont une temperature mini -85.5 et temperature maxi +300.5
        // le dernier capteur a une température mini de -45.5 et maxi de 60.6
        //...

        // Afficher 8 mesures par capteurs (12 lignes de 8 valeurs) et afficher pour chaque capteur s'il est con
forme ou non
        cout << "\n\nCAPTEURS DE TEMPERATURE (MODELE EXTERIEUR) " << endl;
        //...

        // Affichage du nombre de capteurs
        cout << "ETAPE4 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 22)" << endl << endl;

        // Supprimer tous les capteurs
        //...

        // Affichage du nombre de capteurs
        cout << "ETAPE5 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 0)" << endl << endl;

        _getch(); // attente d'appui sur une touche
        return 0; // sortie du programme
}
```