- La suppression des zéros de début fonctionne presque
- La recherche de la plus longue suite de zéros ne fonctionne pas.
- La reconstruction fonctionne quand il y a 8 hextets, mais ne fonctionne pas si on a remplacé la plus longue suite de zéros par "::"

```
/**********************
IPV6.h
**********************/
#pragma once
#include <string>
#include <vector>
using namespace std;

class IPV6
{
protected:

	string adresse;
	bool error;
	void remplacementSuiteDeZero();
	bool TestIP(string adresse);
	static vector<string> split(const string& s, char c);

public:

	vector<string> strHextets;
	void supressionDesZeroDeDebut();
	void reconstructionFinale();

	IPV6(string adr);
	string getAdresse() { return adresse; }
	bool getError() { return error; }
};



/**********************
IPV6.cpp
**********************/
#include "ipv6.h"
#include <string>
#include <vector>
using namespace std;

vector<string> IPV6::split(const string& s, char c)
{
	vector <string> v;
	unsigned int i = 0;
	unsigned int j = s.find(c);
	while (j < s.length())
	{
		v.push_back(s.substr(i, j - i));
		i = ++j;
		j = s.find(c, j);
		if (j >= s.length())
		{
			v.push_back(s.substr(i, s.length()));
			break;
		}
	}
	return v;
}

IPV6::IPV6(string adr)
{
	this->adresse = adr;

}

bool IPV6::TestIP(string adresse)
{
	string impossible = "ghijklmnopqrstuvwxyzGHIJKLMN?OPQRSTUVWXYZ";
	if (adresse.length() != 39)
	{
		return false;
	}
	for (int i = 0; i < impossible.length(); i++)
	{
		for (int j = 0; j < adresse.length(); j++)
		{
			if (impossible.substr(i, 1) == adresse.substr(j, 1))
			{
				return false;
			}
		}
	}
	return true;
}

void IPV6::supressionDesZeroDeDebut()
{
```

```
        int nbzero = 0;
        this->adresse = adresse;
        char c = ':';
        vector<string> v = IPV6::split(adresse, c);
        //Return une chaine décomposée.
        this->strHextets = v;

        for (int i = 0; i < v.size(); i++)
        {
                nbzero = 0;
                {
                        if (v[i].substr(0, 1) == "0")
                        {
                                nbzero = 1;
                                for (int j = 1; j < 4; j++)
                                {
                                        if (v[i].substr(j, 1) == "0")
                                        {
                                                nbzero++;
                                        }
                                        else
                                        {
                                                break;
                                        }

                                }
                        }
                        if (nbzero > 0 && nbzero != 4)
                        {
                                v[i].erase(0, nbzero);

                        }
                }
        }
        this->strHextets = v;
}

void IPV6::remplacementSuiteDeZero()
{
        int chaineZero = 0, nbzero = 0, chaineZeroMax = 0, position = 0;
        vector<string> v = this->strHextets;
        for (int i = 0; i < v.size(); i++)
        {
                nbzero = 0;
                {
                        if (v[i].substr(0, 1) == "0")
                        {
                                nbzero = 1;
                                for (int j = 1; j < 4; j++)
                                {
                                        if (v[i].substr(j, 1) == "0")
                                        {
                                                nbzero++;
                                        }
                                        else
                                        {
                                                chaineZero = 0;
                                                break;
                                        }

                                }
                        }
                        if (v[i].substr(0, 1) != "0")
                        {
                                chaineZero = 0;
                        }
                        if (nbzero == 4)
                        {
                                chaineZero++;
                        }
                        if (chaineZero > chaineZeroMax)
                        {
                                chaineZeroMax = chaineZero;
                                position = i;
                        }
                }
        }
        int Nmax = position - 1 + chaineZeroMax;
        for (int n = position - 1; n < Nmax; n++)
        {
                v[n].erase(0, 4);
        }
        this->strHextets = v;

}
```

Il faut traiter quand même le cas où on a 4 zéros pour en laisser 1

```
if (nbzero == 4)
        v[i].erase(0, 3);
```

pourquoi ce test avec 4 ?

Il faut écrire des commentaires pour que votre code soit compréhensible !
Par exemple, je ne comprends pas la différence entre nbZero et chaineZero

```
void IPV6::reconstructionFinale()
{

        this->supressionDesZeroDeDebut();
        this->remplacementSuiteDeZero();
        string adresse2;
        vector<string> v = this->strHextets;
        vector<string> tmp;
        int position = 100;
        for (int j = 0; j < v.size(); j++)
        {
                if (j == position)
                {
                        adresse2 = adresse2 + ":";
                }
                if (j == 7 && v[6] != "0")
                {
                        adresse2 = adresse2 + v[j];
                }
                else
                {
                        adresse2 = adresse2 + v[j] + ":";
                }
        }
        this->adresse = adresse2;
}


/**********************
main.cpp
**********************/
#include <iostream>
#include "windows.h"
#include <vector>
#include <string>
#include <conio.h>
#include "ipv6.h"

int main()
{
        string IP = "2001:0db8:0000:85a3:0000:0000:ac1f:8001";
        IPV6 monIP(IP);
        monIP.reconstructionFinale();
        cout << monIP.getAdresse();

        cin.ignore();
        return EXIT_SUCCESS;
}
```

Ce test avec j=7 ne fonctionne que si on a 8 hextets donc dans le cas où on n'a pas remplacé la plus longue suite de zéros