```cpp
#include <string>
#include <iostream>
#include "Cible.h"
#include "Collection.h"
#include "Graphe.h"
#include "Log.h"
using namespace std;

static void testLog();
static void testConstructLog();
static void testConstructLog2();

static void testCible();
static void testConstructCible();
static void testAjouterCible();
static void testCompteCible();

static void testCollection();
static void testConstructCollection();
static void testTop10Collection();
static void testTop10CollectionVide();
static void testTop10CollectionE();
static void testTop10CollectionT();
static void testTop10CollectionET();
static void testTop10CollectionEgalite();
static void testTop10CollectionMoins10();

static void testGraphe();
static void testGrapheConstruct();
static void testGrapheConstructE();
static void testGrapheConstructT();
static void testGrapheConstructET();
static void testGrapheGenereFichier();


static const string logGet = "192.168.0.0 - - [08/Sep/2012:11:16:02 +0200] \"GET /
temps / 4IF16.html HTTP / 1.1\" 200 12106 \"http://intranet-if.insa-
lyon.fr/temps/4IF15.html\" \"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1\"";
static const string logPost = "192.168.0.0 - -[08/Sep / 2012:11 : 16 : 07 + 0200]
\"POST /temps/4IF20.html HTTP/1.1\" 200 5185 \"http://intranet-if.insa-
lyon.fr/temps/4IF19.html\" \"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1\"";
static const string logInter = "192.168.0.4 - - [08/Sep/2012:11:19:13 +0200] \"GET
/ SiteWebIF / Intranet - etudiant.php ? ticket = ST - 341667 -
KHlNEzic9e5btb4JQ1Nw - dsi - vm03 HTTP / 1.1\" 302 - \" - \" \"Mozilla / 5.0
(Windows NT 6.1; WOW64) AppleWebKit / 537.1 (KHTML, like Gecko) Chrome /
21.0.1180.89 Safari / 537.1\"";
static const string logsTestCollec = "testCollec.in";
static const string logsTestCollecEgalite = "testCollecEgalite.in";
static const string logsTestCollecVide = "testCollecVide.in";
static const string logsTestCollecMoins10 = "testCollecMoins10.in";



int main()
{
	cout << "Programme de tests" << endl;
```

```
	testLog();
	testCible();
	testCollection();
	testGraphe();
	cout << "Tapez du texte pour quitter" << endl;
	string zzz;
	cin >> zzz;
	return 0;
}



void testCible()    //batterie de tests de la classe Cible
{
	testConstructCible();
	testAjouterCible();
	testCompteCible();
}

void testConstructCible()
{
	cout << "Test du constructeur et destructeur de Cible" << endl;
	Cible *tcible = new Cible();
	delete tcible;
}

void testAjouterCible()
{
	cout << "Test de Cible::Ajouter" << endl;
	Cible *tcible = new Cible();
	tcible->Ajouter(logGet);
	tcible->Ajouter(logPost);
	tcible->Ajouter(logInter);
	delete tcible;
}

void testCompteCible()
{
	cout << "Test de Cible::Compte" << endl;
	Cible *tcible = new Cible();
	tcible->Ajouter(logGet);
	tcible->Ajouter(logGet);
	tcible->Ajouter(logGet);
	tcible->Ajouter(logPost);

	cout << "Comptage des GET -- Reponse attendue : 3" << endl;
	cout << tcible->Compte("GET") << endl;

	cout << "Comptage des GET entre 1 et 2h -- Reponse attendue : 0" << endl;
	cout << tcible->Compte("GET", 1) << endl;

	cout << "Comptage des GET entre 9 et 10h -- Reponse attendue : 3" << endl;
	cout << tcible->Compte("GET", 9) << endl;

	cout << "Comptage des POST -- Reponse attendue : 1" << endl;
	cout << tcible->Compte("POST") << endl;
```

```
	cout << "Comptage des Virsolvy -- Reponse attendue : 0" << endl;
	cout << tcible->Compte("Virsolvy") << endl;

	delete tcible;
}
static void testLog()
{
	cout << "Test de Log" << endl;
	testConstructLog();
	testConstructLog2();
}
static void testConstructLog()
{
	cout << "Test du constructeur 1" << endl;
	Log tLog(logGet);
}
static void testConstructLog2()
{
	cout << "Test du constructeur 2 -- resultat attendu; erreur dans statut" <<
endl;
	Log tLog("192.168.0.0 - -[08/Sep / 2012:11 : 16 : 07 + 0200]");
}

static void testGraphe()
{
	cout << "Test Graphe" << endl;
	testGrapheConstruct();
	testGrapheConstructE();
	testGrapheConstructT();
	testGrapheConstructET();
	testGrapheGenereFichier();
}
static void testGrapheConstruct()
{
	cout << "Test du constructeur" << endl;
	Collection col("test.in");
	Graphe tGraphe(col);
	tGraphe.GenereFichier("test.txt");
}
static void testGrapheConstructE()
{
	cout << "Test du constructeur avec -e" << endl;
	Collection col("test.in");
	Graphe tGraphe(col, true);
	tGraphe.GenereFichier("testE.txt");
}
static void testGrapheConstructT()
{
	cout << "Test du constructeur avec -h" << endl;
	Collection col("test.in");
	Graphe tGraphe(col, false, 11);
	tGraphe.GenereFichier("testH.txt");
}
static void testGrapheConstructET()
{
	cout << "Test du constructeur avec -e -h" << endl;
	Collection col("test.in");
	Graphe tGraphe(col, true,11);
```

```
	tGraphe.GenereFichier("testEH.txt");
}
static void testGrapheGenereFichier()
{
	cout << "Test Graphe::GenereFichier" << endl;

}

static void testCollection()
{
	testConstructCollection();
	testTop10Collection();
	testTop10CollectionMoins10();
	testTop10CollectionVide();
	testTop10CollectionE();
	testTop10CollectionT();
	testTop10CollectionET();
	testTop10CollectionEgalite();
}

static void testConstructCollection()	//test du constructeur de Collection
{
	cout << "Test du constructeur de Collection" << endl;
	cout << "Fichier existant:" << endl;
	Collection tcollec(logsTestCollec);
	cout << "Fichier inexistant:" << endl;
	Collection tcollec2("Virsolvy");
}

static void testTop10Collection() //test de top10 sans option ni cas particulier
{
	cout << "test de Collection::Top10 -- cas general:" << endl;
	Collection tcollec(logsTestCollec);
	tcollec.Top10();
}

static void testTop10CollectionVide()	//test de top10 avec une collection vide
{
	cout << "test de Collection::Top10 -- Collection vide:" << endl;
	Collection tcollec(logsTestCollecVide);
	tcollec.Top10();
}

static void testTop10CollectionE()	//test de l'option e pour Top10
{
	cout << "test de Collection::Top10 -- Option -e:" << endl;
	Collection tcollec(logsTestCollec);
	tcollec.Top10(true);
}

static void testTop10CollectionT()	//test de l'option h pour Top10
{
	cout << "test de Collection::Top10 -- Option -h:" << endl;
	Collection tcollec(logsTestCollec);
	tcollec.Top10(false, 9);
}
static void testTop10CollectionET()		//test des options e et h combinees
pour Top10
```

```
{
	cout << "test de Collection::Top10 -- Option -e -h:" << endl;
	Collection tcollec(logsTestCollec);
	tcollec.Top10(true, 9);
}
static void testTop10CollectionEgalite() //test de Top10, cas où on doit dépasser
10 affichages
                                                                        //pour
cause d'egalité
{
	cout << "test de Collection::Top10 -- cas d'egalite:" << endl;
	Collection tcollec(logsTestCollecEgalite);
	tcollec.Top10();
}
static void testTop10CollectionMoins10() //test de Top10, cas où la collection
comporte
                                                                        //moins
de 10 cibles
{
	cout << "test de Collection::Top10 -- cas moins de 10 cibles:" << endl;
	Collection tcollec(logsTestCollecMoins10);
	tcollec.Top10();
}
```