botany.cpp

```
#include <iostream>
#include <string>
#include "topic.h"
#include "section.h"
#include "document.h"
using namespace std;

int main(){
	Topic botany("Botany");
	Document doc1;
	Document doc2("Votaniki by Theofrastos", &botany);
	Document * docPtr = new Document("Peri fyton by Aristotle", &botany);

	cout << "\n-------------------------- SECTIONS OF A DOCUMENT -------------------------\n";
	Section sec1("Intro", 6); Section sec2("Background", 10);
	Section sec3("Actual contribution", 30); Section sec4("Experiments", 10);
	Section sec5("Fin", 4);
	Section sec2_clone(sec2);
	docPtr->addSection(sec1); docPtr->addSection(sec2); docPtr->addSection(sec2_clone);
	cout << "\n----------------------------- DOCUMENT --------------------------\n";
	docPtr->showMe();
	docPtr->showMeDetails();

	cout << "\n\nOffered by the " << Document::getOwner() << endl;
	cout << "\n----------------------------- KILLED BY DEATH-------------------------\n";
	delete docPtr;

	cout << "\n----------------------------- END OF PROGRAM -------------------------\n";
	return 0;
}
```

topic.h

```
#ifndef _TOPIC_H
#define _TOPIC_H

#include <string>
using namespace std;

class Topic{
 public:
	Topic();
	Topic(const string &aTitle);
	~Topic();
	string getTitle();
 private:
	string title;
};

#endif
```

topic.cpp

```
#include <iostream>
#include <string>
#include "topic.h"
using namespace std;

Topic::Topic(){
	title = "NoTitle"; cout << title << endl;
}

Topic::Topic(const string &aTitle){
	title = aTitle; cout << title << endl;
}

Topic::~Topic(){
	cout << "Topic " << title <<" dies\n";
}

string Topic::getTitle() {return title;}
```

section.h

```
#ifndef _SECTION_H
#define _SECTION_H

#include <iostream>
#include <string>
using namespace std;

class Section{
 public:
	Section(){title = "NONAME"; numPages = 0; cout << "Section " << title << " is born\n";}
	Section(const string &aTitle, const int &pages);		//parameterized cstr
	Section(const Section &aSection);				//copy constructor
	~Section(){ cout << "Section " << title << " dies\n";}	//destructor
	string getTitle(){ return title;}
	int getNumPages(){return numPages;}
 private:
	string title;
	int numPages;
};

#endif
```

section.cpp

```
#include <iostream>
#include <string>
#include "section.h"
using namespace std;

Section::Section(const string &aTitle, const int &pages){
	title = aTitle; numPages=pages;
	cout << "Section " << title << " is born\n";
}

Section::Section(const Section &aSection){
	title = aSection.title; numPages = aSection.numPages;
	cout << "Copy cstr fired\n";
}
```

document.h

```
#ifndef _DOCUMENT_H
#define _DOCUMENT_H

#include <iostream>
#include <string>
#include "topic.h"
#include "section.h"
using namespace std;


class Document{
 public:
	Document();
	Document(const string & aName, const Topic * aTopic);
	~Document(){cout << "Doc " << docName << " dies\n";}
	void addSection(const Section &aSection);
	void computeStats(const double & hoursSpentPerPage, int * totalNumPages, double *
		totalEffortToRead);
	void showMe();
	void showMeDetails();
	static string getOwner(){return owner;}
 private:
	string docName;
	Topic * docTopic;
	Section sections[5];
	int numSections;
	static string owner;
};

#endif
```

document.cpp

```
#include <iostream>
#include <string>
#include "document.h"
using namespace std;

string Document::owner = "Library of Univ. Ioannina";

Document::Document(){
	docName="NoName"; docTopic = NULL; numSections= 0;
	cout <<"Empty Doc generated\n";
}

Document::Document(const string & aName, const Topic * aTopic){
	docName=aName;
	docTopic = const_cast<Topic *>(aTopic);
	numSections = 0;
	cout <<"Doc: "<<docName<<" owned by " <<owner<<" with topic "< docTopic->getTitle() << endl;
}

void Document::addSection(const Section &aSection){
	sections[numSections] = aSection;
	cout << "Section " << sections[numSections].getTitle() << " added\n";
	numSections++;
}

void  Document::computeStats(const  double  &  hoursSpentPerPage,  int  *  totalNumPages,  double  *
                        totalEffortToRead){
	*totalNumPages = 0; *totalEffortToRead = 0;
	for (int i=0; i<numSections; i++)
		*totalNumPages += sections[i].getNumPages();
	*totalEffortToRead = *totalNumPages * hoursSpentPerPage;
}

void Document::showMe(){
	cout << owner << " owns a very nice document titled: " << docName << "\n\twith the topic: "
				<< docTopic->getTitle() << "\n";
}

void Document::showMeDetails(){
	cout << "Sections: " << numSections << "\n";
	for (int i=0; i<numSections; i++){
		cout << sections[i].getTitle() <<"\t" << sections[i].getNumPages() << "\n";
	}
	double hrsPerPage = 0.1; int totalPages = 0; double totalEffort = 0;
	this->computeStats(hrsPerPage, &totalPages, &totalEffort);
	cout << "Number of pages: " << totalPages <<endl;
	cout << "Estimated Effort: " << totalEffort << endl;
}
```

makefile

```
all: botanyExample

botanyExample: botany.cpp topic.o section.o document.o
	g++ -Wall -o botanyExample botany.cpp topic.o section.o document.o

document.o: document.h document.cpp topic.h section.h
	g++ -Wall -c document.cpp

section.o: section.h section.cpp
	g++ -Wall -c section.cpp

topic.o: topic.h topic.cpp
	g++ -Wall -c topic.cpp

clean:
	rm *.exe *.o
```