Technical Structure Summary: There are 10 classes declared across 10 .h files stored in directory *headers* and defined across 10 .cpp files stored in the directory *code*. The main file is called *main.cpp* and the execution file is called *story*, both are stored in the root directory. For ease, I have converted the execution file to a text file called *story.txt* file also in root. The code was written in VSCode so there is a *CMake* file included as well, a *references.txt* file is also included. A JPEG picture of the class structure illustrating their relationships is uploaded separately to the repo for ease.

Simulation Description: I implemented a simulation of Monsters University from admission to graduation, where only **Monsters** are admitted to the university, determined based on their **Monster** attributes. The **Monster** attributes mainly consist of physical attributes which are stored as enumerators with constant values, these are declared in *Monster.h*. The simulation assumes four characters, **MikeMonster**, **SullyMonster**, **RandallMonster** and **Boo** which are derived classes with their particular attributes and methods of the **Monster** class, however have constant predefined **Monster** attributes. To begin with, in *main.cpp* we present these characters through method ***Monster**::describe()*. Importantly, each **Monster** starts of with a default level of 'scareability' represented by the attribute isScary, this is arbitrary, set based off the movie, assuming that **RandallMonster** and **SullyMonster** are scary, but **MikeMonster** and **Boo** are not.

The **CourseStructure** class contains a vector of **BookStack** objects which in turn contains a vector of **Book** objects, hence these classes are related through composition. **CourseStructure** also contains an enumerator object of the three **AcademicYears**. In *main.cpp* a vector of books is initialized with arbitrary **Book** attributes, then the **CourseStructure** constructor is called on this book vector which generates a **BookStack** object for each **AcademicYear**, where the books are evenly distributed across the years. The method ***BookStack**::createStack (vector <**Book**>)* is used in this constructor which also makes use of the method ***BookStack**::push (**Book**)* both defined in the *BookStack.cpp*. Once the **CourseStructure** course variable is initialized in *main.cpp* the method ***Monster**::admit(**CourseStructure**)* determines admission to the course (**CourseStructure** and **Monster** are friend classes) based on the attributes nbArms, nbLegs, nbEyes, hasTail, Skin and if passed then a **Student** object is generated with the inherited attributes from **Monster**. **Boo** is hence screened out of advancing to the **Student** class because she is proven to be human, so an empty instance of **Student** is returned in place.

Once admitted to Monsters University, the goal of the simulation is to earn enough scare points (**Student** attribute) to graduate from Monsters University. In *main.cpp* for each **Student** we call upon the setter functions defined in *Student.cpp*. ***Student**::SetYear()* sets the admission year to 1. ***Student**::SetScare()* which based on the inherited **Monster** attribute of isScary determines the initial number of scare points the **Student** has. Scary students start with non-zero points based on their characteristics, while students that are not naturally scary start of with 0 scare points, they need to learn to be scary and can only gain scare points by reading books in the Scaring genre. Lastly, ***Student**::SetType(bool hard_working)* determines how the **Student** chooses to approach their studies i.e. their reading behaviour, the method sets the values of the vector <bool> books_read which is the length of the total number of books on the course, so we know from the start what reading pattern the student will have during the 3 years. hard_working is arbitrarily defined in *main.cpp* based on the movie: **MikeMonster** is not naturally scary, but he is hardworking so he will be a "Dilligent" student, he will read all the books (books_read set to all true values), **RandallMonster** is scary and hard working then he is an "Average" student, he will read some of the books (books_read values are randomly set) and **SullyMonster** is scary with the most scary feature already (he's large) , he doesn't need to work hard so he is a "Slacker" student, he will read none of the books (books_read set to all false values).

Then in *main.cpp* we go through each of the 3 **AcademicYears** (this process could have been summarised as a custom method defined in *Student.cpp*) For each year we create a vector <**BookStack**> of the books that are required for that **AcademicYear** using ***CourseStructure**:: getBookStacks(**AcademicYear**),* then each student is supposed to ***Student**::learn(**BookStack**)* and is then evaluated with ***Student**::pass()* if they advance to the next **AcademicYear**, these are key functions so we detail further.

For ***Student**::learn(**BookStack**),* if the student is in the first year or has passed the previous year, they can learn. The function iterates through the **BookStack** and reads (***Monster**::read()*) the books based on the students set reading behaviour (books_read). If the student chooses to read the book, using ***Student**::SetScare()* the student's scaring ability (scare) is improved by 0.5 points. If the student has not passed the previous year, they cannot learn.

For ***Student**::pass(),* the method uses a switch statement to handle different cases based on the current academic year of the student. To pass the first year you need scare >=1, to pass the second year you need scare>=2 and to pass the third year you need scare>=3. The simulation is designed such that only the "Average" student (that is scary and is hard working like **RandallMonster**) has a chance at graduating from Monsters University (due to *books_read* being randomly initialised, not on all runs does Randall gather enough scare points to graduate). **SullyMonster** as a "Slacker" student with gifted ability starts off with enough scare points to advance to third year, but because he never reads, he cannot graduate, he always fails the third year. **MikeMonster** as a "Dilligent" student with no natural ability, reads every book each year, but it's not enough to compensate for the fact that he is just not scary.

Lastly, all students are initialised as **MUGraduate**, derived class of **Student**, to test whether a student is eligible to graduate from MU University, using ***MUGraduate**::graduate(),* the condition being that the student had to pass all 3 years based on the **Student** attribute vector<bool> year_passes.