Name: Soham Thaker SID: 011-748-159 Email: [sdthaker1@myseneca.ca](mailto:sdthaker1@myseneca.ca)

Firstly, I leaned what are constructors and their uses. Constructors’ sole purpose is to initialize the instance of a class to some default value. At the time of a creation of an object, if no constructor is present, compiler will insert a default, no argument constructor with no code within that constructor and will invoke the constructor automatically and initialize the object if there is any code present otherwise leave the object with garbage values in it. Since the object would have garbage values at the moment of creation, constructors are very helpful in initializing an instance of a class by setting the object to safe empty state. A programmer cannot call the constructor since they are automatically invoked by the compiler at the time of creation of an object. If at least one constructor of any form is present in the program, then the compiler will not insert a default constructor. A constructor does not have any return types and it prototype takes the form “TypeName();” for a default, no argument constructor. In the implementation file, its definition takes the form “TypeName::TypeName() {}” for a default, no argument constructor. We can also overload a constructor by passing multiple arguments. The compiler matches the parameter list of a constructor in its prototype against the object’s declaration and its passed arguments at the moment of creation and invokes the constructor accordingly. An overloaded constructor takes the form “TypeName(int height, double diameter);“ and its definition takes the form “TypeName::TypeName(int height, double diameter) {}”.

Secondly, I learnt what are destructors and their uses. A destructor’s sole purpose is to destroy the object right before it goes out of scope. Destructors are very helpful in deallocating dynamic memory so that we can make sure that there are no memory leaks in the program because it is invoked just before the object is supposed to die. If a class has a resource, and we are to deallocate that memory, a destructor is a perfect place to deallocate it since it is invoked automatically just before the object is supposed to die. All we need to do is insert the code for deallocating dynamic memory and destructor will take care of it automatically. A destructor cannot be overloaded, doesn’t have a return type cannot be called by a programmer. Its prototype takes the form, “~TypeName();” and definition takes the form, “TypeName::~TypeName() {}”. Compiler adds a default, no argument destructor, without any logic within itself if no destructor is declared.

Third, I learnt the usage of \*this in a member function. Every member function has a current object that it operates its logic upon which is an implicit parameter known as ‘this’ object. \*this is basically a pointer that points to the current object so that it can access the attributes of current instance and the logic inside the function can be operated upon those values. At the end of the function, the ‘this’ pointer gets deleted along with any local variables which are all allocated statically as a part of the clean up process.