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

First, I learnt the usage of Rule of 3 which includes a copy constructor, copy assignment operator and a destructor. Copy constructor copies the data from an existing instance to a newly created instance whereas copy assignment operator copies the data from an existing instance to an existing instance. Every class that has resource instance pointers, need to be safely copied into another object at the time of creation or at the time of assignment, otherwise memory leak would happen since compiler does a shallow copy by default which means that variables of an instance are copied into another object, and if the source object has resources the compiler assigns the address of those resources to the current object’s resources. Once the scope ends, the data is deleted for source object since the attributes contained the address where the data is stored and complier calls the destructor for that object(if no code exists in the destructor, it will result in memory leak) which deletes the data stored at source object and results in data loss for the current object as well and creates a memory leak when we try to access the data stored at the current object later on in the program. We need deep copy mechanism to perform this kind of operation. To do this, we can create a copy constructor which takes a constant reference of an existing object as its source parameter. The copy constructor is called when the client code creates the object by initializing it to an existing object, copies an object by value and returns an object by value from a function call. Its logic performs a shallow copy on all non resource instance pointers, allocates memory for each resource and copies data from an existing object to the newly created object. A copy assignment operator accepts a constant reference of an existing object as its source parameter and returns an object by reference. Its logic checks for self assignment, shallow copies all the non resource instance pointers into the destination variables, deallocates any previously allocated memory on the resource variables on the current object, allocates new memory to the resources associated with the current object and finally copes all the resource data from source object to resources of current object. Last but not least, a destructor must be defined that deletes all the resource instances every time when an object goes out of scope and the memory needs to be freed. We can also prevent shallow and deep copying on user-defined types. To do that we write “= delete” after the prototype of a copy constructor and copy assignment operator.

Besides, I also learnt how to overload the extraction and insertion operators as free helper operators which helps cascade a compound type just like a primitive type, example, “cout << A” or “cin >> A” where A is an instance of a compound type. The extraction operator overload accepts a reference of an istream’s instance and a reference of a compound type instance as two of its parameters and returns a reference of an istream’s instance. Same goes with an insertion operator overload except, it accepts a reference of an ostream’s instance as its first parameter and returns a reference of an ostream’s instance.

Furthermore, I learnt how to open a file to output the data onto a file from an object of compound type or how to read the data from a file and store it into an object declared and defined within a program. To create a file object that outputs data to a file, we must create an object of type ofstream, whereas to create a file object that reads data from a file into a user-defined type, we must create an object of type ifstream. To create objects of both types we must include the system header file called fstream where the definition and declaration of both types are present. The objects of ofstream and ifstream work similarly like ostream and istream. If we want to output a data stored in an instance variable to a file connected to the ofstream object first, we must create an object of oftream where its constructor is overloaded with a c-style null terminated string. We can pass the string, which is the file name at the moment of creation, or we can call open() method onto an existing ofstream object and pass the file name as an argument to that function. Once the object is connected to a file we must check if the connection is open. To check a connection, we call the is\_open() method on the file object which returns true or false depending on the established connection. Finally, we can use the insertion operator to insert the data stored in an instance variable and write that data onto the file directly just like cout where the data is outputted to the console. Ifstream and ofstream objects have a bool operator overload which helps if the data extracted or inserted was invalid. If it is invalid, it flags their objects to be unusable until we clear the status just like cin and cout. The process for reading data from a file is very similar except we must create the object of an ifstream class.

Also, I learnt the usage of C++ string class whose sole purpose is to extract an unknown number of characters from the user or a file. To accept the data as input occurs character by character at run time, a C++ string class performs a dynamic memory allocation with the help of a helper function called getline that accepts the file stream or standard input stream object as its first parameter, a string objects as its second and a delimiter as its third which is optional, by default set to a new line. It also has 2 member functions to convert the string stored into its object to a C-style null-terminated string, length() that returns the numbers of characters stored into the string and c\_str() that returns an address of a C-style null terminated version of the string. It’s logic usually has 5 steps, starting with declaring a string object to accept the input, extracting the input using getline function and store the extracted data into the string object, querying the length of the string stored into string object to obtain the amount of memory required, allocating the memory dynamically for a local C-style null-terminated char array in a program, copying the data from the string object into the newly allocated dynamic array and finally deallocating the memory when it needs to be freed.