**Answer the questions in Exercise A in the following table and post it into the D2L**

|  |  |
| --- | --- |
| **Program output and its order** | **Your explanation (why and where is the cause for this output)** |
| **constructor with int argument is called.** | It is called at line 16 in exAmain. The statement, Mystring c = 3 is interpreted by the compiler as a call to the constructor Mystring::Mystring(int n) |
| **default constructor is called.**  **default constructor is called.** | It is called at line 22 in exAmain. The statement, Mystring x[2] is interpreted by the compiler as two calls to the default constructor Mystring::Mystring() |
| **constructor with char\* argument is called.** | It is called at line 26 in exAmain. The statement, Mystring \*z = new Mystring(“4”) is interpreted by the compiler as a call to the constructor Mystring(const char \*s) |
| **copy constructor is called.**  **copy constructor is called.** | It is called at line 28 in exAmain. The statement, x[0].append(\*z).append(x[1]) is interpreted by the compiler as two calls to the copy constructor Mystring(const Mystring &source), the first copying the Mystring object z is pointing to, and the second copying the Mystring object the second element of the pointer array x is pointing to. |
| **destructor is called.**  **destructor is called.** | It is called at line 28 in exAmain. In the statement x[0].append(\*z).append(x[1]) for the method Mystring &append(const Mystring other) the Mystring object is passed by value, which creates an extra copy (other) on each call. The compiler is called twice to delete those extra copies. |
| **copy constructor is called.** | It is called at line 30 in exAmain. The statement Mystring mars = x[0] is interpreted as a call to the copy constructor Mystring(const *Mystring* &*source*), copying in mars over the Mystring object the first element of x points to. |
| **assignment operator called.** | It is called at line 32 in exAmain. The statement x[1] = x[0] is interpreted by the compiler as a call to the assignment operator *Mystring* &operator=(const *Mystring* &*rhs*). |
| **constructor with char\* argument is called.**  **constructor with char\* argument is called.** | It is called at lines 34 and 36 in exAmain. The statements Mystring jupiter(“White”) and arr[0] = new Mystring(“Yellow”) are interpreted by the compiler as calls to the constructor Mystring(const char \*s). |
| **destructor is called.**  **destructor is called.**  **destructor is called.**  **destructor is called.**  **destructor is called.** | It is called 4 times when // BLOCK ENDS HERE is reached. The compiler interprets that the block of code is done so the destructor  ~Mystring() is automatically called to delete the dynamically allocated charsM of the Mystring objects that are now considered out of scope, i.e. jupiter, mars, x[1], and x[0] (in that order). The last call happens at line 41, the destructor is called for delete ar[0]. |
| **constructor with char\* argument is called.** | It is called at line 43. The compiler interprets Mystring d = "Green" as a call to the Mystring(const char \**s*) constructor. |
| **Program terminated successfully.** | It is called at line 45. When we reach cout << "\nProgram terminated successfully." << endl |
| **destructor is called.**  **destructor is called** | It is called at line 47, at the end of our program. The destructor is called twice to remove the dynamically allocated charsM of the remaining Mystring objects, i.e., d and c (in that order). |