PS # 8

Deadline: 17.05.2020 at 23:50

Hand-in Policy:

Create a cpp file for each part of the assignment and save the screenshot while these parts are running.

Your assignment should consist of 3 cpp files and 3 screenshots.

Zip all files of the assignment. Your zip files should look like this

Ps8_studentnumber_name_surname.zip

1)

You would like to verify the credentials of a user for your system. Listed next is a class named Security , which authenticates a user and password. (Note that this example is really not very secure. Typically passwords would be encrypted or stored in a database.)

```
class Security
{
public:
        static int validate(string username, string password);
};
// This subroutine hard-codes valid users and is not
// considered a secure practice.
// It returns 0 if the credentials are invalid,
// 1 if valid user, and
// 2 if valid administrator
int Security::validate(string username, string password)
{
        if ((username=="abbott") && (password=="monday")) return 1;
        if ((username=="costello") && (password=="tuesday")) return 2;
        return 0;
}
```

Break this class into two files, a file with the header *Security.h* and a file with the implementation *Security.cpp*.

Next, create two more classes that use the Security class by including the header file. The first class should be named Administrator and contain a function named Login that returns true if a given username and password have administrator

clearance. The second class should be named User and contain a function named Login that returns true if a given username and password have either user or administrator clearance. Both the User and Administrator classes should be split into separate files for the header and implementation.

Finally, write a main function that invokes the Login function for both the User and Administrator classes to test if they work properly. The main function should be in a separate file. Be sure to use the #ifndef directive to ensure that no header file is included more than once.

2)

This Programming Project explores how the unnamed namespace works. Listed are snippets from a program to perform input validation for a username and password.

The code to input and validate the username is in a separate file than the code to input and validate the password.

File header user.cpp:

```
namespace Authenticate
{
	void inputUserName( )
	{
		do
		{
		cout << "Enter your username (8 letters only)" << endl;
		cin >> username;
		} while (!isValid( ));
	}
	string getUserName( )
	{
		return username;
	}
}
```

Define the username variable and the isValid( ) function in the unnamed namespace so the code will compile. The isValid( ) function should return true if username contains exactly eight letters. Generate an appropriate header file for this code.

Repeat the same steps for the file password.cpp, placing the password variable and the isValid( ) function in the unnamed namespace. In this case, the isValid( ) function should return true if the input password has at least eight characters including at least one non-letter:

File header password.cpp:

```
namespace Authenticate
{
	void inputPassword( )
	{
		do
		{
		cout << "Enter your password (at least 8 characters " <<
			"and at least one non-letter)" << endl;
		cin >> password;
		} while (!isValid( ));
	}
	string getPassword( )
	{
		return password;
	}
}
```

At this point, you should have two functions named isValid( ) , each in different unnamed namespaces. Place the following main function in an appropriate place. The program should compile and run.

```
int main( )
{
	inputUserName( );
	inputPassword( );
	cout << "Your username is " << getUserName( ) <<
		" and your password is: " <<
		getPassword( ) << endl;
	return 0;
}
```

Test the program with several invalid usernames and passwords.

3)

This exercise is intended to illustrate namespaces and separate compilation in your development environment. You should use the development environment you regularly use in this course for this exercise. In a file f.h , place a declaration of void f( ) in namespace A . In a file g.h , place a declaration of void g( ) in namespace A . In files f.cpp and g.cpp , place the definitions of void f( ) and void g( ) , respectively. Place the definitions of void f( ) and void g( ) in namespace A . The functions can do anything you want, but to keep track of execution include something like

cout << "*Function_Name called*" << endl;

where *Function_Name* is the name of the particular function. In another file, main.cpp , put your main function, #include the minimum collection of files to provide access to the names from namespace A . In your main function call the functions f then g . Compile, link, and execute using your development environment. To provide access to names in namespaces, you may use local using declarations such as

using std::cout;

or use local using directives such as

using namespace std;

inside a block, or qualify names using the names of namespaces, such as std::cout .

You may not use global namespace directives such as the following which are not in a block and apply to the entire file:

using namespace std;

Of course you must handle namespace A and function names f and g , in addition to possibly std and cout .

After doing this, write a one page description of how to create and use namespaces and separate compilation in your environment.