**A - Lists**

Description:

This program creates two lists and makes use of all the functions and operators requested for lists available in the C++ STL. Assistance was taken from cpluslus.com in order to learn how to utilize these list functions and how to utilize them properly. While researching about traversing through lists in C++, I found that iterators could be avoided if I was using C++11. However, I avoided that method for compatibility concerns. An even function was also made that is utilized for the remove\_if function. Lists are similar to vectors, expect vectors are good when the type of sequential container used is of little matter. Lists are good for when the intent is to insert or remove elements in the structure at positions other than the end.

Output Example:

Begin function implemented. Assign function: 50 50 50 Insert function: 50 50 50 4 Back function: 4 Push\_back function: 50 50 50 4 7 Push\_front function: 6 50 50 50 4 7 Empty function: The list is not empty. Pop\_back function: 6 50 50 50 4 Pop\_front function: 50 50 50 4 Sort function: 4 50 50 50 = function implemented. mylist2: 4 50 50 50 Clear function on mylist implemented. Remove function implemented. Remove value 50: 4 Remove\_if function implemented. Remove even numbers. Size function: 0 elements in mylist1 and mylist 2.

**B - Stacks**

Description:

This program creates two stacks and makes use of all the functions and operators requested for stacks available in the C++ STL. Assistance was taken from cpluslus.com in order to learn how to utilize these stack functions and how to utilize them properly. I found that there were far fess available functions for stacks as there were for lists. The stacks work as expected being a Last In First Out data structure.

Output Example:

Push function: 5 8 6 7 Pop function: 8 6 7 Top function: 8 = function implemented. mystack2 = mystack1: 8 6 7 Empty function: mystack2 is not empty. Size function for mystack2: 3

**C - Queues**

Description:

This program creates two regular queues and one priority queue. It makes use of all the functions and operators requested for queues available in the C++ STL. Assistance was taken from cpluslus.com in order to learn how to utilize these queues functions and how to utilize them properly. Like stacks, there are far less functions available for queues than there are for lists. The syntax for the queue functions is just as well very similar to the function for queues. The primary difference is that queues operate in a First In First Out manner. The top function was also requested, and that function is available for priority queues.

Output Example:

Push function: 13 15 12 14 Pop function: 15 12 14 Front function: 15 Back function: 14 = function implemented. myqueue2 = myqueue1: 15 12 14 Empty function: myqueue2 is not empty. Size function for myqueue2: 3 Top function: 15

**D - Namespaces**

Description:

This program makes use of the code given and makes a few changes to display the variable and functions from the namespaces. Assistance was taken from cplusplus.com to learn more about namespaces and what they are used for and how to call them. The scope operator :: is used in this case.

Output Example:

You can call variables and functions outside the namespace using ::, the scope operator.

from orange from blue 1 2