Lab 03 - Arrays & Linked Lists

Direction: Submit typed work in the Labs directory of your github repository and/or as an attachment on Google classroom under the accurate Lab03 assessment. All submissions should have their appropiate extensions.

Part A: In class

Your objective is to modify the accompanying file **lab03.cpp** by defining the following functions

- ☐ A void function named `ArrayFill()` whose header is

```
template <typename T>
void ArrayFill(Array<T>& data,const T& value)
```

  It assigns *value* to every element of *data*.

- ☐ A void function named `SubArray()` whose header is

```
template <typename T>
void SubArray(Array<T>& data,Array<T>& subdata,ulong a,ulong b)
```

  If *a* and *b* are both valid indices of *data* [they are both between 0 and the size of *data*], the function will resize *subdata* so that its size is equal to 1 more than the distance between *a* and *b*, and then, it will assign the values of *data* between *a* and *b* inclusively to *subdata*. If *a* and *b* are not both valid, the function does nothing. Do not assume *a* is less than *b*.

- ☐ A ulong function named `DelimitedSearch()` whose header is

```
template <typename T>
ulong DelimitedSearch(Array<T>& data,const T& delimit,const T& target)
```

  It returns the index of the first element whose value is equal to *target* that precedes the first instance of *delimit* in *data*. If *delimit* is not found in *data* or an instance of *target* cannot be found before the first instance of *delimit*, the function returns the size of *data*.

Warning: adding or removing any libraries in the file will result in a 0. Only the modified file will be an acceptable submission.

Part B: Take home

Your objective is to write a complete file that includes the library **Node.h** and defines the bool function named `Monotonic()` whose header is

```
bool Monotonic(Node<int>* head)
```

Given that *head* is pointing to a doubly linked list, the function returns true if either the list is empty or the linked list is monotonically increasing; otherwise, it returns false. A linked list, a, is montonically increasing if

$$data(a_i) \leq data(a_j)$$

for all $i < j$ where i and j positions of the nodes in the linked list.