WAYNE STATE UNIVERSITY
COLLEGE OF ENGINEERING

Computer Science Department

CSC 2201: Computer Science II – Lab
Lab3

Description:

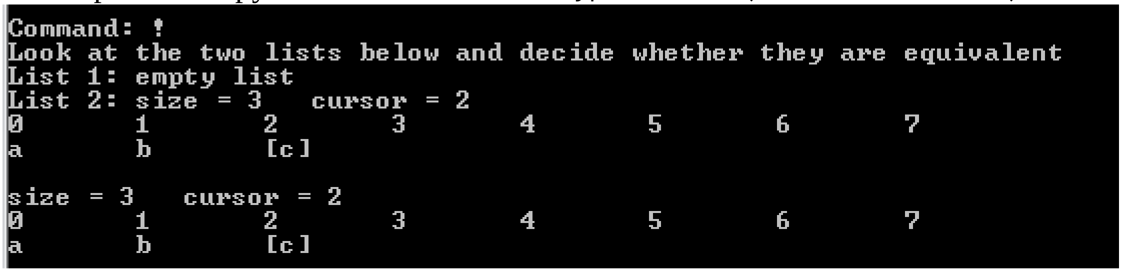
You will implement your own List ADT, the ListArray class.

Goals:

Learn how to use C++ classes to implement List abstract data type (ADT).

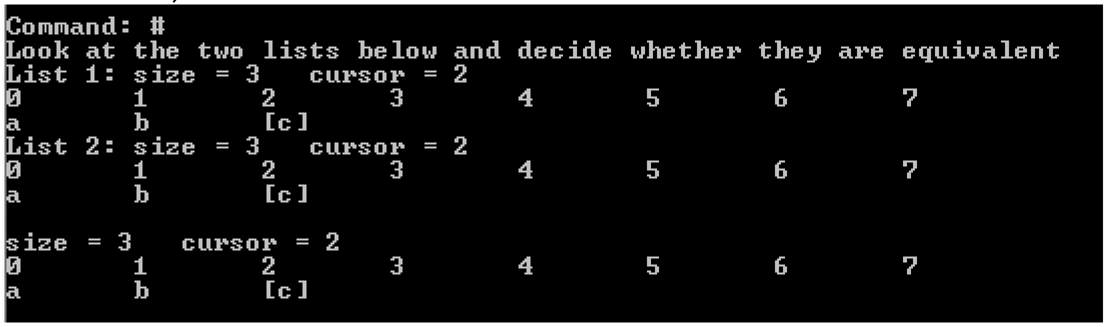
Where to Start:

1. Download Lab3.zip from Blackboard
2. Unzip the file
3. Open the solution in Microsoft Visual Studio
4. Start Implementing the operations of the ListArray class (only the body of methods)
    4.1. Implement default constructor: `List<DataType>:: List (int maxNumber )`
    4.2. Implement copy constructor: `List<DataType>:: List (const List &source)`



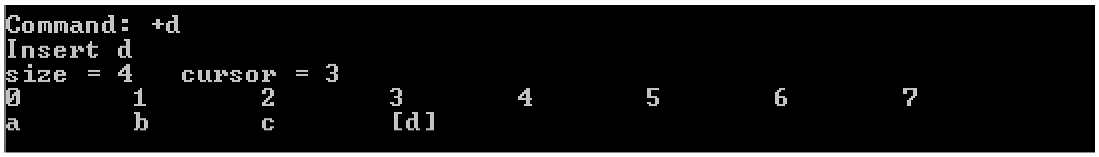
```
Command: !
Look at the two lists below and decide whether they are equivalent
List 1: empty list
List 2: size = 3   cursor = 2
0       1       2       3       4       5       6       7
a       b       [c]

size = 3   cursor = 2
0       1       2       3       4       5       6       7
a       b       [c]
```

    4.3. Implement operator =: `List<DataType>& List<DataType>::operator = ( const List &source)`



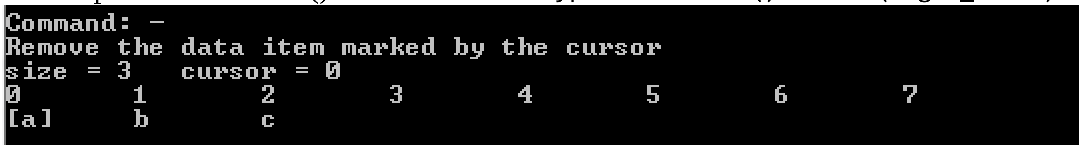
```
Command: #
Look at the two lists below and decide whether they are equivalent
List 1: size = 3   cursor = 2
0       1       2       3       4       5       6       7
a       b       [c]
List 2: size = 3   cursor = 2
0       1       2       3       4       5       6       7
a       b       [c]

size = 3   cursor = 2
0       1       2       3       4       5       6       7
a       b       [c]
```

    4.4. Implement destructor: `List<DataType>:: ~List ()`
    4.5. Implement insert (): `void List<DataType>:: insert ( const DataType &newDataItem ) throw ( logic_error )`



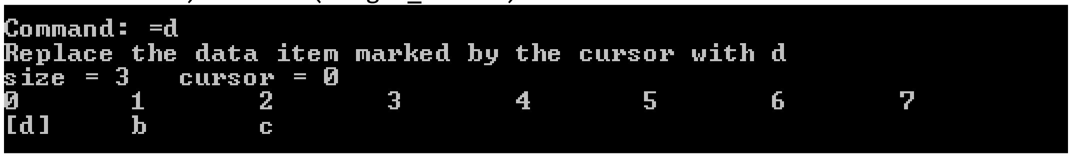
```
Command: +d
Insert d
size = 4   cursor = 3
0       1       2       3       4       5       6       7
a       b       c       [d]
```

4.6. Implement remove (): `void List<DataType>:: remove () throw (logic_error)`



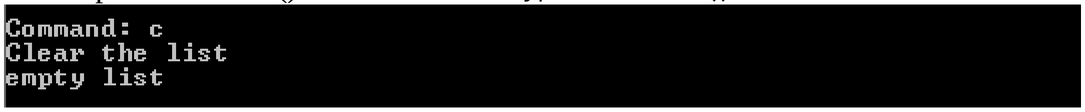
```
Command: -
Remove the data item marked by the cursor
size = 3   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c
```

4.7. Implement replace (): `void List<DataType>:: replace ( const DataType &newDataItem ) throw  ( logic_error )`



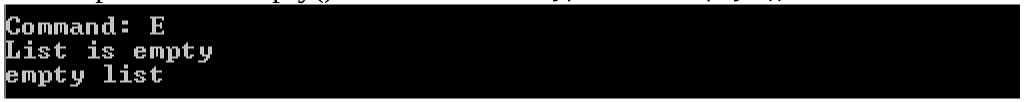
```
Command: =d
Replace the data item marked by the cursor with d
size = 3   cursor = 0
0       1       2       3       4       5       6       7
[d]     b       c
```

4.8. Implement clear (): `void List<DataType>:: clear ()`



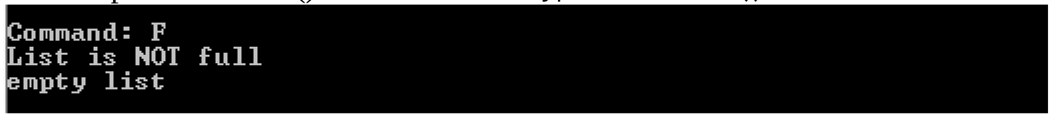
```
Command: c
Clear the list
empty list
```

4.9. Implement isEmpty():`bool List<DataType>:: isEmpty () const`



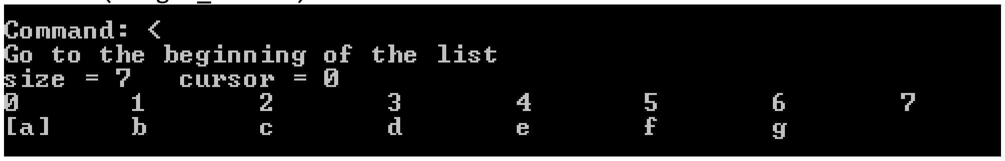
```
Command: E
List is empty
empty list
```

4.10. Implement isFull(): `bool List<DataType>:: isFull () const`



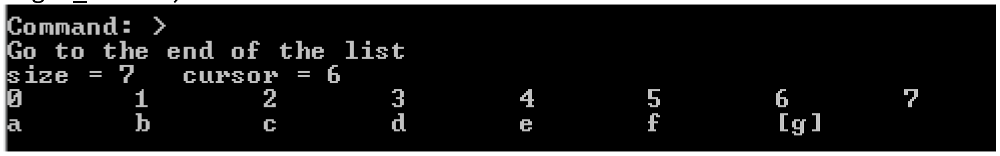
```
Command: F
List is NOT full
empty list
```

4.11. Implement gotoBeginning ():`void List<DataType>:: gotoBeginning () throw  ( logic_error )`



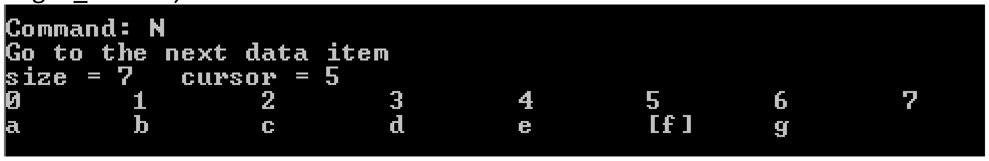
```
Command: <
Go to the beginning of the list
size = 7   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c       d       e       f       g
```

4.12. Implement gotoEnd ():`void List<DataType>:: gotoEnd () throw ( logic_error )`



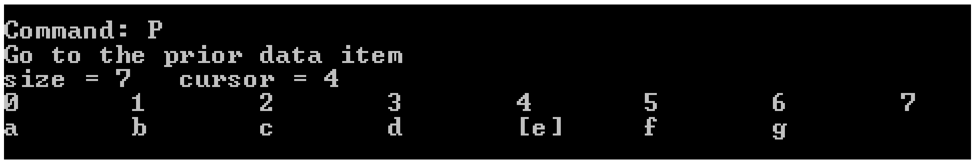
```
Command: >
Go to the end of the list
size = 7   cursor = 6
0       1       2       3       4       5       6       7
a       b       c       d       e       f       [g]
```

4.13. Implement gotoNext ():`void List<DataType>:: gotoNext () throw ( logic_error )`



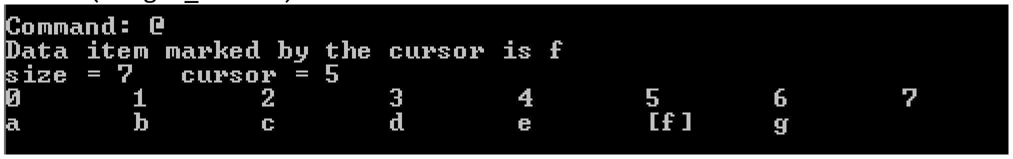
```
Command: N
Go to the next data item
size = 7   cursor = 5
0       1       2       3       4       5       6       7
a       b       c       d       e       [f]     g
```

4.14. Implement gotoPrior ():`void List<DataType>:: gotoPrior () throw ( logic_error )`



```
Command: P
Go to the prior data item
size = 7   cursor = 4
0        1        2        3        4        5        6        7
a        b        c        d        [e]      f        g
```

4.15. Implement getCursor (): `DataType List<DataType>:: getCursor () const throw ( logic_error )`

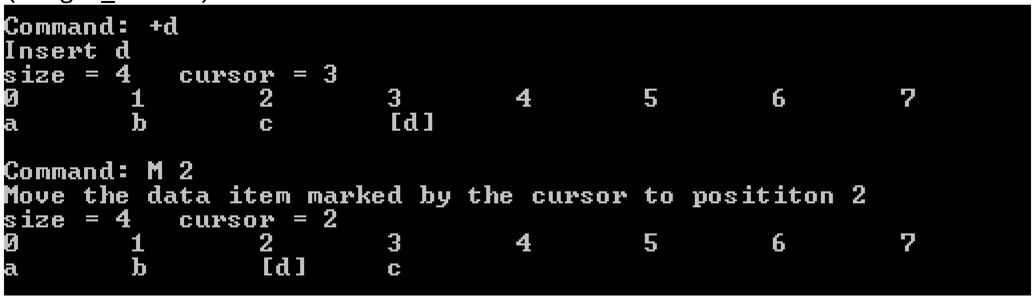


```
Command: @
Data item marked by the cursor is f
size = 7   cursor = 5
0        1        2        3        4        5        6        7
a        b        c        d        e        [f]      g
```

4.16. Implement showStructure ():`void List<DataType>:: showStructure ()const`

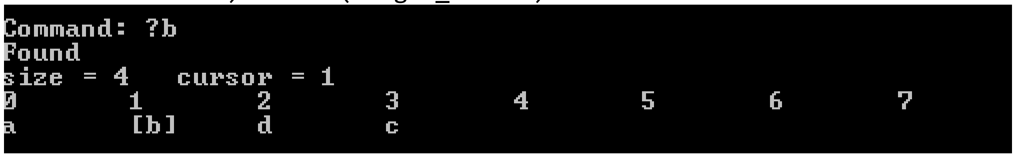
5. Compile your implementation of the List ADT in the file ListArray.cpp and the test program in the file test3.cpp.
6. Test your implementation using the program in the file test3.cpp.
7. Implement more methods:

   7.1. Implement moveToNth ():`void List<DataType>:: moveToNth (int n ) throw ( logic_error )`



```
Command: +d
Insert d
size = 4   cursor = 3
0        1        2        3        4        5        6        7
a        b        c        [d]

Command: M 2
Move the data item marked by the cursor to posititon 2
size = 4   cursor = 2
0        1        2        3        4        5        6        7
a        b        [d]      c
```

   7.2. Implement find ():`bool List<DataType>:: find (const DataType &searchDataItem ) throw ( logic_error )`



```
Command: ?b
Found
size = 4   cursor = 1
0        1        2        3        4        5        6        7
a        [b]      d        c
```

8. Activate test 2 and test 3 in the program test3.cpp by changing the definition of LAB3_TEST2 from 0 to 1 in config.h to test moveToNth () and LAB3_TEST3 from 0 to 1 to test find ().

Create a Zip file of your solution:

1. Right click on your solution in Solution Explorer
2. Click on “Open Folder in File Explorer”
3. Go one level up in file explorer
4. Right click on your solution folder
5. Add it to archive by creating a zip file

Upload the zipped file on Blackboard:

1. Go to Blackboard

2. Click on this course (CSC 2201: Computer Science II Lab)
3. Go to the folder “Labs”
4. Click on the “Lab3_Work” assignment
5. Upload your zipped file