WAYNE STATE UNIVERSITY

COLLEGE OF ENGINEERING

Computer Science Department

CSC 2201: Computer Science II – Lab
Lab4

Description:

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

Goals:

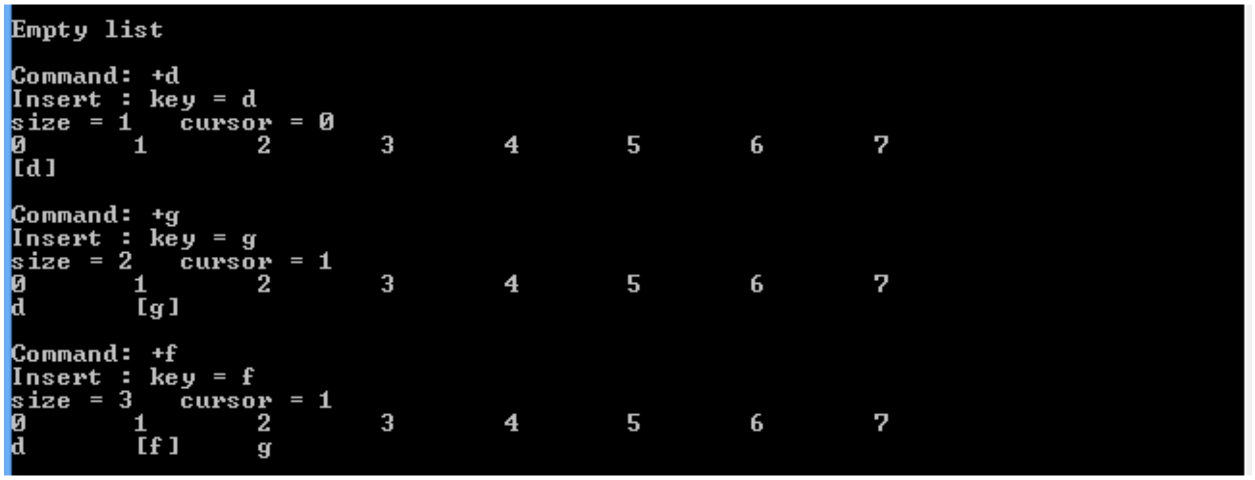
Learn how to use C++ classes to implement List abstract data type (ADT).

Where to Start:

1. Download Lab4.zip from Blackboard
2. Unzip the file
3. Open the solution in Microsoft Visual Studio
4. Start Implementing the operations of the ListArray class (from Lab3)
    4.1. Implement default constructor: `List<DataType>:: List (int maxNumber )`
    4.2. Implement copy constructor: `List<DataType>:: List (const List &source)`
    4.3. Implement operator =: `List<DataType>& List<DataType>::operator = ( const List &source)`
    4.4. Implement destructor: `List<DataType>:: ~List ()`
    4.5. Implement insert (): `void List<DataType>:: insert ( const DataType &newDataItem ) throw ( logic_error )`
    4.6. Implement remove (): `void List<DataType>:: remove () throw (logic_error)`
    4.7. Implement replace (): `void List<DataType>:: replace ( const DataType &newDataItem ) throw  ( logic_error )`
    4.8. Implement clear (): `void List<DataType>:: clear ()`
    4.9. Implement isEmpty():`bool List<DataType>:: isEmpty () const`
    4.10. Implement isFull(): `bool List<DataType>:: isFull () const`
    4.11. Implement gotoBeginning ():`void List<DataType>:: gotoBeginning () throw  ( logic_error )`
    4.12. Implement gotoEnd ():`void List<DataType>:: gotoEnd () throw ( logic_error )`
    4.13. Implement gotoNext ():`void List<DataType>:: gotoNext () throw ( logic_error )`
    4.14. Implement gotoPrior ():`void List<DataType>:: gotoPrior () throw ( logic_error )`
    4.15. Implement getCursor (): `DataType List<DataType>:: getCursor () const throw ( logic_error )`
    4.16. Implement showStructure ():`void List<DataType>:: showStructure ()const`

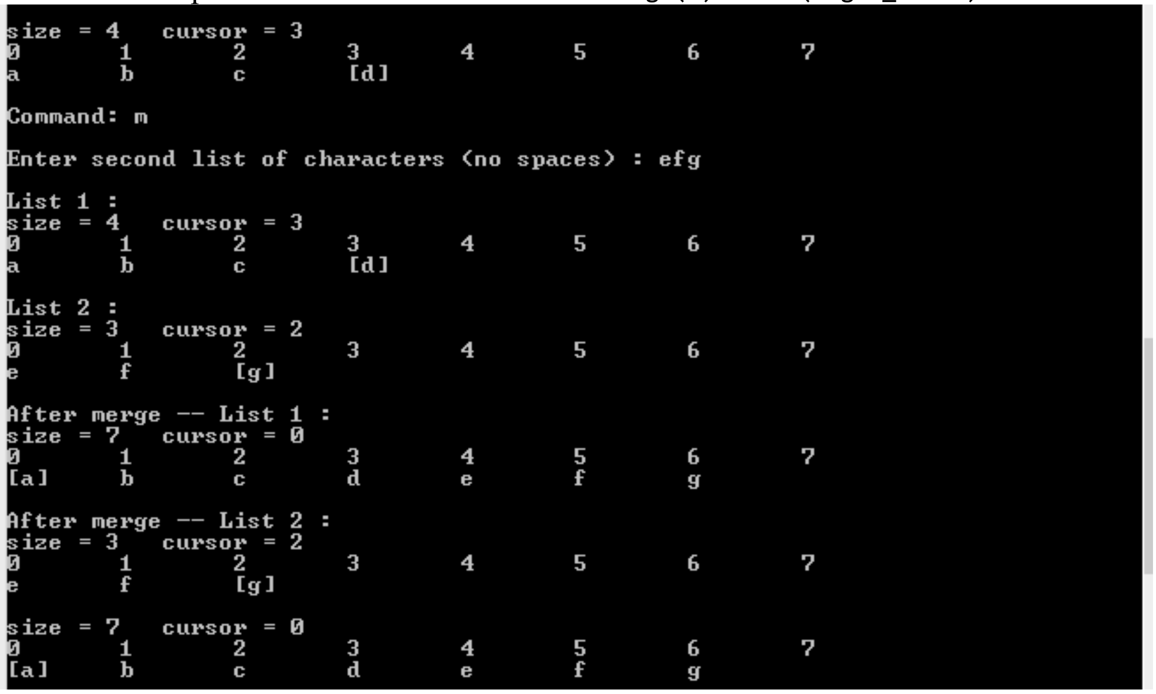
5. Start Implementing the operations of the OrderedList class
    5.1. Implement default constructor: `OrderedList<…>::OrderedList (int maxNumber)`
    5.2. Implement: `void OrderedList<…>::insert(…) throw(logic_error)`
    5.3. Implement: `bool OrderedList<…>::retrieve(…)`
    5.4. Implement: `void OrderedList<…>::replace(…) throw(logic_error)`
    5.5. Implement: `void OrderedList<…>::showStructure () const`
    5.6. Implement: `bool OrderedList<…>::binarySearch(…)`

The outputs should be like Lab3, except when you insert a new element, it will be sorted.



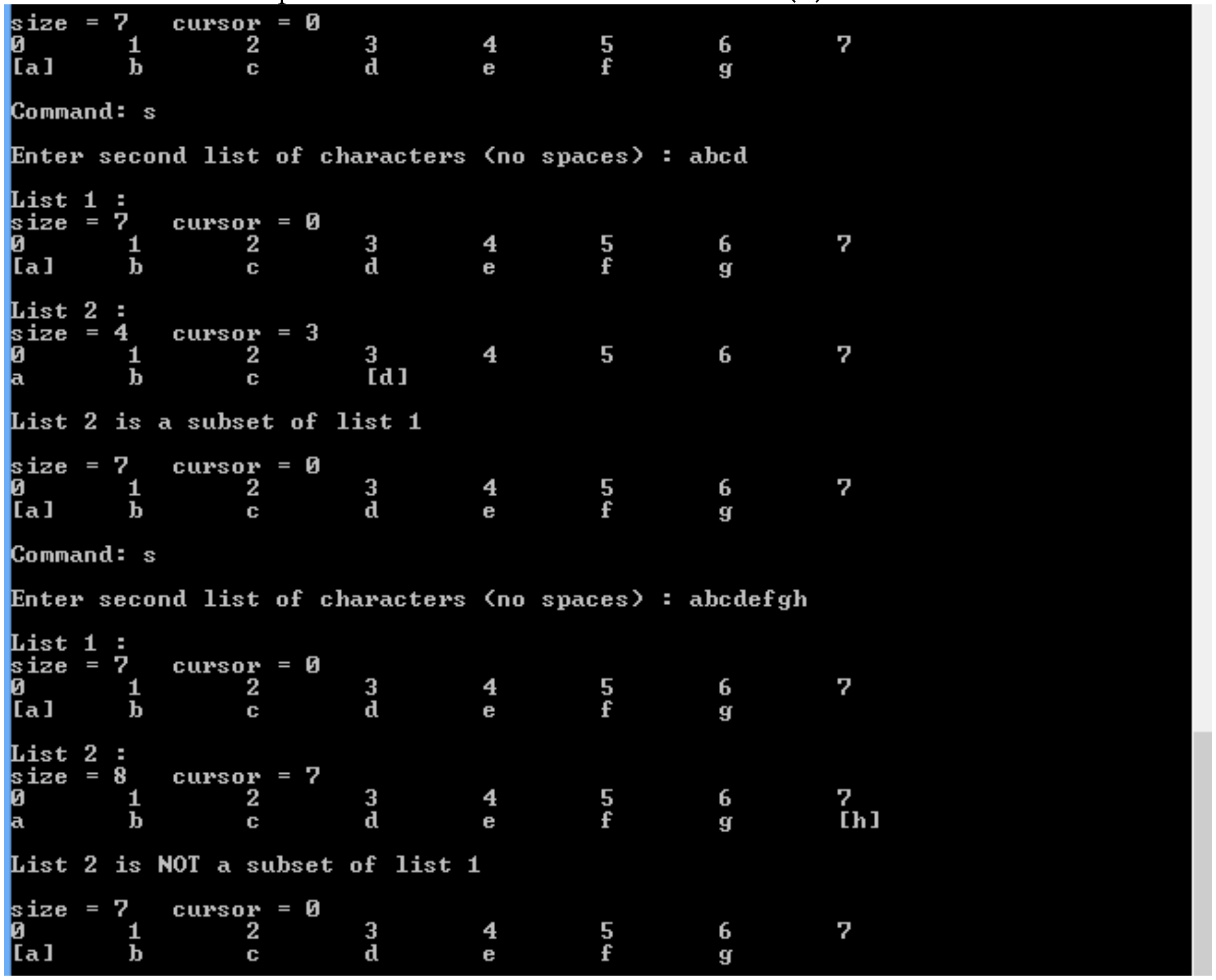
```
Empty list

Command: +d
Insert : key = d
size = 1   cursor = 0
0       1       2       3       4       5       6       7
[d]

Command: +g
Insert : key = g
size = 2   cursor = 1
0       1       2       3       4       5       6       7
d       [g]

Command: +f
Insert : key = f
size = 3   cursor = 1
0       1       2       3       4       5       6       7
d       [f]     g
```

6. Compile your implementation of the List ADT in the file ListArray.cpp and the test program in the file test4.cpp.
7. Test your implementation using the program in the file test4.cpp.
8. Implement more methods:
    8.1. Implement: `void OrderedList<…>::merge(…) throw(logic_error)`



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

Command: m

Enter second list of characters (no spaces) : efg

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

List 2 :
size = 3   cursor = 2
0       1       2       3       4       5       6       7
e       f       [g]

After merge -- List 1 :
size = 7   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c       d       e       f       g

After merge -- List 2 :
size = 3   cursor = 2
0       1       2       3       4       5       6       7
e       f       [g]

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

8.2. Implement: `bool OrderedList<…>::isSubset(…)`



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

Command: s

Enter second list of characters (no spaces) : abcd

List 1 :
size = 7   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c       d       e       f       g

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

List 2 is a subset of list 1

size = 7   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c       d       e       f       g

Command: s

Enter second list of characters (no spaces) : abcdefgh

List 1 :
size = 7   cursor = 0
0       1       2       3       4       5       6       7
[a]     b       c       d       e       f       g

List 2 :
size = 8   cursor = 7
0       1       2       3       4       5       6       7
a       b       c       d       e       f       g       [h]

List 2 is NOT a subset of list 1

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

9. Test your implementation using the program in the file test4.cpp.

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 "Lab4_Work" assignment
5. Upload your zipped file