

Problem Set 3

Submission Instruction and Requirement:

Due date: 2:30pm 26th April 2018

- 1) Create respective folders for C++ source codes of each question and zip the folders of tasks you have attempted in one zip file. **Do not** include any Microsoft Visual Studio solution files in your submission.
- 2) Name the file in the pattern of studentid.yourname.ps3.zip OR. studentid.yourname.ps3.rar
- 3) Write a report (at least 1 page for each task attempted, including screenshots, if you've attempted both tasks your report should be at least 2 pages long) on the codes you have created and include the report in the zip/rar file mentioned in 2) with screenshots of successful running of your codes. If the code does not work as expected, please provide justifications.
- 4) The codes should be neat and be well-commented.
- 5) Your code should be workable and without any error, warning message, infinite loop or any malicious function.
- 6) Submit the zip/rar file to Blackboard on time.

Task 1 (14 marks)

```
template<class DataType>
class DoublyLinkedListNode
{
public:
    typedef DoublyLinkedListNode<DataType> Node;

private:
    DataType fValue;
    Node* fNext;
    Node* fPrevious;

    DoublyLinkedListNode()
    {
        fValue = DataType();
        fNext = &NIL;
        fPrevious = &NIL;
    }

public:
    static Node NIL;

    DoublyLinkedListNode(const DataType& aValue);

    void prepend(Node& aNode);
    void append(Node& aNode);
    void remove();

    const DataType getValue() const;
    const Node* getNext() const;
    const Node* getPrevious() const;
};

template<class DataType>
DoublyLinkedListNode<DataType> DoublyLinkedListNode<DataType>::NIL;
```

The specification of class `DoublyLinkedListNode` from Lab 6 is given above. Based on your implementation of `DoublyLinkedListNode` class in Lab 6, develop an application that can construct a sorted doubly linked list from the unsorted double array below.

```
double lData[] = { 37.3, 20.6, 138.9, 70.0, 55.9, 25.9, 144.4, 66.9, 112.6, 106.7, 134.2, 129.5 };
```

The application will print a comparison of each node (previous, current, and next) in the non-sorted doubly linked list vs each node (previous, current, and next) in the sorted (in ascending order) doubly linked list.

Requirement: You must make use of the `DoublyLinkedListNode.h` from Lab 6 in your solution. You may choose to separate the implementation to `DoublyLinkedListNode.cpp`. You are allowed to add new methods, classes or iterator to support your solution (not compulsory).

Task 1 Output Example Screenshot:

```
C:\Windows\system32\cmd.exe

All Nodes without Sorting:
Prev      Current      Next
<<NULL>,  37.3,        20.6>
<37.3,     20.6,        138.9>
<20.6,     138.9,        70>
<138.9,    70,         55.9>
<70,       55.9,        25.9>
<55.9,     25.9,        144.4>
<25.9,     144.4,        66.9>
<144.4,    66.9,        112.6>
<66.9,     112.6,       106.7>
<112.6,    106.7,       134.2>
<106.7,    134.2,       129.5>
<134.2,    129.5,       <NULL>>

All Nodes after Sorting:
Prev      Current      Next
<<NULL>,  20.6,        25.9>
<20.6,     25.9,        37.3>
<25.9,     37.3,        55.9>
<37.3,     55.9,        66.9>
<55.9,     66.9,        70>
<66.9,     70,         106.7>
<70,       106.7,       112.6>
<106.7,    112.6,       129.5>
<112.6,    129.5,       134.2>
<129.5,    134.2,       138.9>
<134.2,    138.9,       144.4>
<138.9,    144.4,       <NULL>>
Press any key to continue . . .
```

Task 2 (6 marks)

Similar to Task 1, develop an application that construct a sorted doubly linked-list (based on alphabetical order) from the following strings of name:

"Emma", "Zack", "Wade", "Liam", "Kyle", "Mason", "Fiona", "Sam", "Ava",
"Mike", "Diana", "Paul", "Ryan", "Aidan", "Beth", "Noel", "Tina", "Harry",
"Cyril", "Jean"

The application will print the sorted names from the doubly linked-list.

Requirement: You must make use of the `DoublyLinkedListNode.h` from Lab 6 in your solution. You may choose to separate the implementation to `DoublyLinkedListNode.cpp`. You are allowed to add new methods, classes or iterator to support your solution (not compulsory).

Hint: You may use the `compare()` from string class to compare two strings.

Task 2 Output Example Screenshot:

```
C:\Windows\system32\cmd.exe

All Nodes without Sorting:
Prev      Current      Next
<<NULL>,  Alan,        Zack>
<Alan,    Zack,        Wade>
<Zack,    Wade,        Liam>
<Wade,    Liam,        Kyle>
<Liam,    Kyle,        Mason>
<Kyle,    Mason,        Fiona>
<Mason,    Fiona,        Sam>
<Fiona,    Sam,        Ava>
<Sam,     Ava,        Mike>
<Ava,     Mike,        Diana>
<Mike,    Diana,        Paul>
<Diana,    Paul,        Ryan>
<Paul,    Ryan,        Beth>
<Ryan,    Beth,        Cyril>
<Beth,    Cyril,        Noel>
<Cyril,    Noel,        Tina>
<Noel,    Tina,        Harry>
<Tina,    Harry,        Emma>
<Harry,    Emma,        Jean>
<Emma,    Jean,        <NULL>>

All Nodes after Sorting:
Prev      Current      Next
<<NULL>,  Alan,        Ava>
<Alan,    Ava,        Beth>
<Ava,     Beth,        Cyril>
<Beth,    Cyril,        Diana>
<Cyril,    Diana,        Emma>
<Diana,    Emma,        Fiona>
<Emma,    Fiona,        Harry>
<Fiona,    Harry,        Jean>
<Harry,    Jean,        Kyle>
<Jean,     Kyle,        Liam>
<Kyle,    Liam,        Mason>
<Liam,    Mason,        Mike>
<Mason,    Mike,        Noel>
<Mike,    Noel,        Paul>
<Noel,    Paul,        Ryan>
<Paul,    Ryan,        Sam>
<Ryan,    Sam,        Tina>
<Sam,     Tina,        Wade>
<Tina,    Wade,        Zack>
<Wade,    Zack,        <NULL>>
Press any key to continue . . . -
```