Problem Set 2

<u>Submission Instruction and Requirement:</u>

Due date: 12th April 2019

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.ps2.zip
   OR.
   studentid.yourname.ps2.rar
3) Write a report (at least 1 page for each task attempted, including screenshots, if you've attempted 3 tasks your report should be at least 3 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) You 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 (12 marks)

Based on this ListNodeIterator class declaration, you are required to fully implement ListNodeIterator (i.e. all methods in this class) in a file named "ListNodeIterator.cpp".

ListNodeTemplate.h:

```
template <class T>

class ListNode
{
public:
	T fData;
	ListNode* fNext;
	ListNode(const T& aData, ListNode* aNext = (ListNode*)0)
	{
		fData = aData;
		fNext = aNext;
	}
 };
```

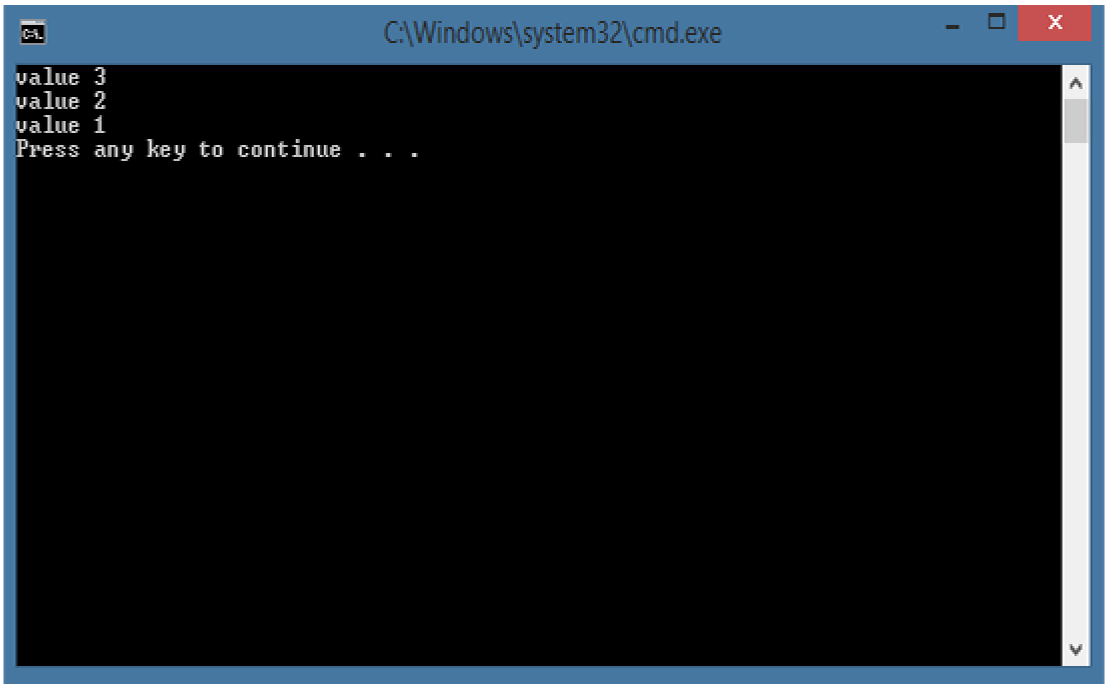
ListNodeIterator.h:

```
#include "ListNodeTemplate.h"

template<class T>
class ListNodeIterator
{
private:
	ListNode<T>* fNode;
public:
	typedef ListNodeIterator<T> Iterator;
	ListNodeIterator(ListNode<T>* aNode);
	const T& operator*() const;
	Iterator& operator++();
	Iterator operator++(int);
	bool operator==(const Iterator& aOther) const;
	bool operator!=(const Iterator& aOther) const;
	Iterator end();

};
```

You can test your iterator with the following main function (Copy this to your **main.cpp**):

```
#include<iostream>
#include "ListNodeIterator.cpp"

using namespace std;

int main()
{
	typedef ListNode<int> IntegerNode;

	IntegerNode One(1);
	IntegerNode Two(2, &One);
	IntegerNode Three(3, &Two);

	for (ListNodeIterator<int> iter(&Three); iter != iter.end(); ++iter)
	{
		cout << "value " << *iter << endl;
	}

	return 0;
}
```

Console output for Task 1:



```
value 3
value 2
value 1
Press any key to continue . . .
```

You should get the same output with your code if your ListNodeInterator.cpp is implemented correctly.

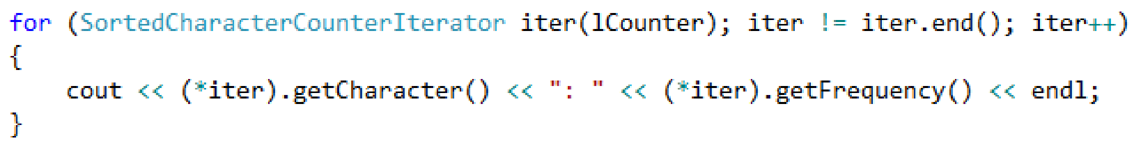
Task 2 (12 marks)
Based on the task in Lab 5, fully implement class SortedCharacterCounterIterator below in "SortedCharacterCounterIterator.cpp" using a sorting algorithm other than bubble sort (e.g. insertion sort, quick sort, merge sort, heap sort etc.), sort according to *frequency* in descending order (from highest frequency to lowest frequency).

Please state clearly in your report which sorting algorithm you have chosen and provide a screenshot of the output as proof.

SortedCharacterCounterIterator.h:

```
#include "CharacterCounter.h"
#include "FrequencyMap.h"

class SortedCharacterCounterIterator {
private:
	FrequencyMap fMaps[256];
	int fIndex;
public:
	SortedCharacterCounterIterator(CharacterCounter& aCounter);

	SortedCharacterCounterIterator(const SortedCharacterCounterIterator&
aCounterIt, int aIndex);

	// return current frequency map
	const FrequencyMap& operator*() const;
	SortedCharacterCounterIterator& operator++(); // prefix
	SortedCharacterCounterIterator operator++(int); // postfix (extra unused
argument)
	bool operator==(const SortedCharacterCounterIterator& aOther) const;
	bool operator!=(const SortedCharacterCounterIterator& aOther) const;
	SortedCharacterCounterIterator begin() const;
	SortedCharacterCounterIterator end() const;
};
```

Task 3 – Optional Task (6 marks)
SortedCharacterCounterIterator in Task 2 is a forward iterator. Using this iterator, you will be able to print the character frequencies in descending order (highest frequency to lowest frequency) using the following for loop:



```
for (SortedCharacterCounterIterator iter(lCounter); iter != iter.end(); iter++)
{
    cout << (*iter).getCharacter() << ": " << (*iter).getFrequency() << endl;
}
```

For this task, in a separate project, improve the SortedCharacterCounterIterator by turning it into a bidirectional iterator that can also be used to print the character frequencies in ascending order (lowest frequency to highest frequency). Revise the iterator test in main.cpp to showcase that your iterator can move backward.

Please state clearly in your report what changes you have made to the forward iterator in task 2 to convert it to a bidirectional iterator and provide a screenshot of the output as proof.