Swinburne University of Technology

School of Science, Computing and Engineering Technologies

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Assignment number and title: 2, Iterators
Due date: Monday, April 17, 2023, 10:30
Lecturer: Dr. Markus Lumpe

Your name: Brandy Dan **Your student ID:** 103864526

Check Tutorial	Tues 08:30	Tues 10:30	Tues 12:30 BA603	Tues 12:30 ATC627	Tues 14:30	Wed 08:30	Wed 10:30	Wed 12:30	Wed 14:30	Thurs 08:30	Thurs 10:30
						X					

Marker's comments:

Problem	Marks	Obtained
1	16	
2	22	
3	92	
Total	130	

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener: _____

```
#include "CharacterMap.h"
#include <cstddef>

CharacterMap::CharacterMap(unsigned char aCharacter , int aFrequency ) noexcept:
	fCharacter(aCharacter), fFrequency(aFrequency)
{}

void CharacterMap:: increment() noexcept
{
	fFrequency++;
}


void CharacterMap::setCharacter(unsigned char aCharacter) noexcept
{
	fCharacter = aCharacter;
}

bool CharacterMap::operator<(const CharacterMap& aOther) const noexcept
{
	return (*this).fFrequency < aOther.fFrequency;
}

unsigned char CharacterMap::character() const noexcept
{
	return fCharacter;
}

size_t CharacterMap::frequency() const noexcept
{
	return fFrequency;
}
```

```
#include "CharacterCounter.h"
#include <iostream>
#include <algorithm>



CharacterCounter::CharacterCounter() noexcept:
	fTotalNumberOfCharacters(0),fCharacterCounts()
{}

void CharacterCounter::count(unsigned char aCharacter) noexcept
{
	bool a = true;
	for (int i = 0; i < 256; i++)
	{
		if (aCharacter == fCharacterCounts[i].character())
		{
			fCharacterCounts[i].increment();
			a = false;
		}
	}

	if (a == true)
	{
		fCharacterCounts[fTotalNumberOfCharacters].setCharacter(aCharacter);
		fCharacterCounts[fTotalNumberOfCharacters].increment();
		fTotalNumberOfCharacters++;
	}

	for (int i = 1; i < 256; i++)
	{
		CharacterMap key = fCharacterCounts[i];

		int j = i - 1;
		while (j >= 0 && fCharacterCounts[j].character() > key.character())
		{
			std::swap(fCharacterCounts[j + 1], fCharacterCounts[j]);
			j = j - 1;
		}
		fCharacterCounts[j + 1] = key;
	}


}


const CharacterMap& CharacterCounter::operator[](unsigned char aCharacter) const noexcept
{
	return fCharacterCounts[aCharacter];
}
```

```
#include "CharacterFrequencyIterator.h"
#include <iostream>
#include <algorithm>


void CharacterFrequencyIterator::mapIndices() noexcept
{
	for (int i = 0; i < 256; i++)
	{
		fMappedIndices[i] = static_cast<unsigned char>(i);
	}



	for (int i = 1; i < 256; i++)
	{
		unsigned char key = fMappedIndices[i];

		int j = i - 1;
		while (j >= 0 && (*fCollection)[fMappedIndices[j]] < (*fCollection)[key])
		{
			std::swap(fMappedIndices[j + 1], fMappedIndices[j]);
			j = j - 1;
		}
		fMappedIndices[j + 1] = key;

	}
}
CharacterFrequencyIterator::CharacterFrequencyIterator(const CharacterCounter* aCollection) noexcept :
	fCollection(aCollection), fIndex(0), fMappedIndices()
{
	mapIndices();
}

const CharacterMap& CharacterFrequencyIterator::operator*() const noexcept
{
	return (*fCollection)[fMappedIndices[fIndex]];
}

CharacterFrequencyIterator& CharacterFrequencyIterator::operator++() noexcept
{
	fIndex++;
	return *this;
}

CharacterFrequencyIterator CharacterFrequencyIterator :: operator++(int) noexcept
{
	CharacterFrequencyIterator old = *this;
	++(*this);
	return old;

}
```

```
bool CharacterFrequencyIterator:: operator==(const CharacterFrequencyIterator& aOther) const noexcept
{
	return fCollection == aOther.fCollection && fIndex == aOther.fIndex;

}

bool CharacterFrequencyIterator::operator!=(const CharacterFrequencyIterator& aOther) const noexcept
{
	return !(*this == aOther);

}

CharacterFrequencyIterator CharacterFrequencyIterator::begin() const noexcept
{
	CharacterFrequencyIterator copy = *this;

	copy.fIndex = 0;
	copy.mapIndices();
	return copy;


}

CharacterFrequencyIterator CharacterFrequencyIterator::end() const noexcept
{
	CharacterFrequencyIterator copy = *this;
	int result = 0;
	for (int i = 0; i < 256; i++)
	{
		if ((*fCollection)[static_cast<unsigned char>(i)].frequency() != 0)
		{
			result++;
		}
	}
	copy.fIndex = result;
	return copy;

}
```