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: Sartaj Khan **Your student ID:** 103509694

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
								●			

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:____________

```cpp


#include "CharacterMap.h"



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 fFrequency < aOther.fFrequency;
}

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

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


```

```cpp


#include "CharacterCounter.h"


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


void CharacterCounter::count(unsigned char aCharacter) noexcept {
    fCharacterCounts[aCharacter].setCharacter(aCharacter);
    fCharacterCounts[aCharacter].increment();
    fTotalNumberOfCharacters++;
}

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

```

```cpp


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


void CharacterFrequencyIterator::mapIndices() noexcept {
    for (int i = 0; i < 256; i++) {
        fMappedIndices[i] = (*fCollection)[i].character();
    }
    size_t i = 1;

    while (i < 256)
    {
        size_t j = i;

        while (j > 0 && std::less<CharacterMap>{}((*fCollection)[fMappedIndices[j - 1]], (*fCollection)[fMappedIndices[j]]))
        {
            std::swap(fMappedIndices[j - 1], fMappedIndices[j]);
            j--;
        }

        i++;
    }
}


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


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

CharacterFrequencyIterator& CharacterFrequencyIterator::operator++() noexcept {
    fIndex++;
    CharacterFrequencyIterator result = *this;
    if ((*result).frequency() == 0) fIndex = 256;
    return result;
}

CharacterFrequencyIterator CharacterFrequencyIterator::operator++(int) noexcept
```

```
  {
    CharacterFrequencyIterator old = *this;
    ++(*this);
    return old;
}

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

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

CharacterFrequencyIterator CharacterFrequencyIterator::begin() const noexcept {
    CharacterFrequencyIterator result = *this;
    result.fIndex = 0;
    return result;
}

CharacterFrequencyIterator CharacterFrequencyIterator::end() const noexcept {
    CharacterFrequencyIterator result = *this;
    result.fIndex = 256;
    return result;
}

```