# 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:** Ekrar Efaz **Your student ID:** 103494172

| 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: _____

```cpp
//
//  CharacterMap.cpp
//  Assignment2
//
//  Created by Ekrar Efaz on 4/4/23.
//

#include "CharacterMap.h"

// constructor with default arguments, it becomes a default constructor
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;
}
```

```cpp
//
//  CharacterCounter.cpp
//  Assignment2
//
//  Created by Ekrar Efaz on 4/4/23.
//

#include "CharacterCounter.h"

CharacterCounter::CharacterCounter() noexcept : fTotalNumberOfCharacters(0)
{
    for (int i = 0; i < 256; ++i)
    {
        fCharacterCounts[i] = CharacterMap(i,0);
    }
}

// Increment the frequency of the given character
void CharacterCounter::count(unsigned char aCharacter) noexcept
{
    fCharacterCounts[aCharacter].setCharacter(aCharacter);
    fCharacterCounts[aCharacter].increment();
    ++fTotalNumberOfCharacters;
}

// Return the CharacterMap object corresponding to the given character
const CharacterMap& CharacterCounter::operator[](unsigned char aCharacter) const noexcept
{
    return fCharacterCounts[aCharacter];
}
```

```cpp
//
//  CharacterFrequencyIterator.cpp
//  Assignment2
//
//  Created by Ekrar Efaz on 12/4/23.
//

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

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

    // stable insertion sort algorithm
    /*

     */
    for (int i = 1; i < 256; ++i){
        unsigned char currentCharacter = fMappedIndices[i]; // start with a character at the current index
        int previousCharacterIndex = i - 1; // set the previous character index
        while(previousCharacterIndex>=0 && fCollection->operator[](fMappedIndices[previousCharacterIndex]).frequency() < fCollection->operator[](currentCharacter).frequency()){
            // continue the loop until you go through all the previous elements until the first onw to keep swapping
            std::swap(fMappedIndices[previousCharacterIndex + 1], fMappedIndices[previousCharacterIndex]);
            -- previousCharacterIndex;
        }
    }
}

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

const CharacterMap& CharacterFrequencyIterator::operator*() const noexcept
{
    return fCollection->operator[](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;
    return copy;
}

/*
I started off with implemented fIndex=256 which resulted in showing a lot more characters which had zero frequency. My solution suggests
that I find the last non-zero character index and then set end to one past that. I use a while loop for this implementation
*/

CharacterFrequencyIterator CharacterFrequencyIterator::end() const noexcept {
    CharacterFrequencyIterator copy = *this;
    int i = 255; // start from the last character index
    while (i >= 0 && copy.fCollection->operator[](copy.fMappedIndices[i]).frequency() == 0) {
        --i; // search for last non-zero frequency character
    }
    copy.fIndex = i + 1; // set index to one past the last non-zero frequency character
    return copy;
}
```