

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: Md Redwan Ahmed Zawad _____ **Your student ID:** 103501849 _____

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: _____

```
#include "CharacterMap.h"
```

```
CharacterMap::CharacterMap(unsigned char aCharacter , int aFrequency ) noexcept:
```

```
    fFrequency(aFrequency),  
    fCharacter(aCharacter)
```

```
{  
}
```

```
void CharacterMap::increment() noexcept
```

```
{  
    fFrequency++;  
}
```

```
void CharacterMap::setCharacter(unsigned char aCharacter) noexcept
```

```
{  
    fCharacter = aCharacter;  
}
```

```
unsigned char CharacterMap::character() const noexcept
```

```
{  
    return fCharacter;  
}
```

```
size_t CharacterMap::frequency() const noexcept
```

```
{  
    return fFrequency;  
}
```

```
bool CharacterMap::operator<(const CharacterMap& aOther) const noexcept
```

```
{  
    return fFrequency < aOther.fFrequency;  
}
```

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

```
CharacterCounter::CharacterCounter() noexcept
```

```
{  
    size_t i = 0;  
    while (i < 256)  
    {  
        fCharacterCounts[i].setCharacter(static_cast<unsigned char>(i));  
        i++;  
    }  
    fTotalNumberOfCharacters = 0;  
}
```

```
void CharacterCounter::count(unsigned char aCharacter) noexcept
```

```
{  
  
    fCharacterCounts[aCharacter].increment();  
    fTotalNumberOfCharacters++;  
  
}
```

```
const CharacterMap& CharacterCounter::operator[](unsigned char aCharacter) const noexcept
```

```
{  
    return fCharacterCounts[aCharacter];  
}
```

```

#include "CharacterFrequencyIterator.h"
#include<algorithm>
void CharacterFrequencyIterator::mapIndices() noexcept
{
    for (size_t i = 0; i < 256; i++)
    {
        fMappedIndices[i]=static_cast<unsigned char>(i);
    }
    size_t i = 0;
    while (i < 256)
    {
        size_t j = i+1;
        while (j > 0 && (*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()

{
    mapIndices();
}

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

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

CharacterFrequencyIterator CharacterFrequencyIterator:: operator++(int)noexcept
{
    CharacterFrequencyIterator old = *this;
    ++(*this);
    if ((*fCollection)[fMappedIndices[fIndex]].frequency() == 0)
    {
        fIndex = 256;
    }
    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 Result = *this;
    Result.fIndex = 0;
    return Result;
}

CharacterFrequencyIterator CharacterFrequencyIterator::end() const noexcept
{
    CharacterFrequencyIterator Result = *this;
    Result.fIndex = 256;
    return Result;
}
}1

```