Swinburne University Of Technology

Faculty of Information and Communication Technologies

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures & Patterns
Assignment number and title: 3 – Design Patterns and 12 Bit I/O
Due date: May 13, 2024, 10:30
Lecturer: Dr. Markus Lumpe

Your name: Vu Duc Tran **Your student id:** 104175614

Marker's comments:

Problem	Marks	Obtained
1	138	
Total	138	

Extension certification:

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

Signature of Convener:_____

ifstream12.cpp

```
//
//  ifstream12.cpp
//  problem3
//
//  Created by Vu Duc Tran on 9/5/2024.
//


// COS30008, Problem Set 3, 2024

#include "ifstream12.h"

ifstream12::ifstream12(const char* aFileName, size_t aBufferSize)
    : fBuffer(new std::byte[aBufferSize]), fBufferSize(aBufferSize),
fByteIndex(0), fBitIndex(7), fByteCount(0)
{
    reset();
    open(aFileName);
}

ifstream12::~ifstream12()
{
    close();
    delete[] fBuffer;
}

void ifstream12::reset()
{
    fByteIndex = 0;
    fByteCount = 0;
    fBitIndex = 7;
}

void ifstream12::fetch_data()
{
    if (fByteCount == 0) {
        fIStream.read(reinterpret_cast<char*>(fBuffer), fBufferSize);
        fByteCount = fIStream.gcount();
        fByteIndex = 0;
        fBitIndex = 7;
    }
}

std::optional<size_t> ifstream12::readBit()
{
    if (fByteCount == 0)
        fetch_data();

    if (fByteCount == 0)
        return std::nullopt;

    std::byte lByte = fBuffer[fByteIndex] & (std::byte{ 1 } << fBitIndex);
    size_t bitValue = std::to_integer<size_t>(lByte) > 0 ? 1 : 0;

    if (--fBitIndex < 0) {
```

```
        fBitIndex = 7;
        ++fByteIndex;
        --fByteCount;
    }

    return bitValue;
}

void ifstream12::open(const char* aFileName)
{
    // Close any open file first
    close();
    fIStream.open(aFileName, std::ios::binary);
}

void ifstream12::close()
{
    fIStream.close();
}

bool ifstream12::isOpen() const
{
    return fIStream.is_open();
}

bool ifstream12::good() const
{
    return fIStream.good();
}

bool ifstream12::eof() const
{
    return fByteCount == 0;
}

ifstream12& ifstream12::operator>>(size_t& aValue)
{
    aValue = 0;
    int bitPosition = 0;
    for (int i = 11; i >= 0; --i) {
        auto bit = readBit();
        if (!bit.has_value()) break;
        aValue |= static_cast<size_t>(*bit) << bitPosition;
        bitPosition++;
    }
    return *this;
}
```