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: Nur E Siam **Your student id:** 103842784

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:_____

```cpp
// Assignment 3
// created by Nur E Siam

#include "ifstream12.h"
#include <cassert>
#include <limits>
#include <iostream>
using namespace std;

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

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

void ifstream12::open(const char* aFileName)
{
    assert(!isOpen());
    fIStream.open(aFileName, std::ifstream::binary);
    if (isOpen())
    {
        fetch_data();
    }
}

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

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

```
50  }
51
52  bool ifstream12::good() const
53  {
54      return fIStream.good() && (fByteCount > 0 || !fIStream.eof());
55  }
56
57  bool ifstream12::eof() const
58  {
59      return fByteCount == 0;
60  }
61
62  void ifstream12::fetch_data()
63  {
64      fIStream.read(reinterpret_cast<char*>(fBuffer), fBufferSize);
65      fByteCount = fIStream.gcount();
66      fByteIndex = 0;
67      fBitIndex = 7;
68  }
69
70  void ifstream12::reset()
71  {
72      for (size_t i = 0; i < fBufferSize; i++)
73      {
74          fBuffer[i] = std::byte{ 0 };
75      }
76      fByteCount = 0;
77      fByteIndex = 0;
78      fBitIndex = 7;
79  }
80
81  ifstream12& ifstream12::operator>>(size_t& aValue)
82  {
83      aValue = 0;
84      for (size_t i = 0; i < 12; i++)
85      {
86          std::optional<size_t> bitOpt = readBit();
87          if (!bitOpt.has_value())
88          {
89              break;
90          }
91          if (bitOpt.value() == 1)
92          {
93              aValue |= static_cast<size_t>(bitOpt.value()) << i;
94          }
95      }
96      return *this;
97  }
98
```

```
 99 std::optional<size_t> ifstream12::readBit()
100 {
101     if (fByteCount == 0)
102     {
103         if (fIStream.eof())
104         {
105             return std::nullopt;
106         }
107         fetch_data();
108     }
109 
110     if (fByteCount == 0)
111     {
112         return std::nullopt;
113     }
114     std::byte lByte = fBuffer[fByteIndex] & (std::byte{ 1 } << fBitIndex);
115     size_t bitValue = std::to_integer<size_t>(lByte);
116 
117     fBitIndex--;
118     if (fBitIndex < 0)
119     {
120         fBitIndex = 7;
121         fByteIndex++;
122         fByteCount--;
123     }
124 
125     if (bitValue == 0)
126     {
127         return 0;
128     }
129     else
130     {
131         return 1;
132     }
133 }
134 
```