COS30008 Semester August 2024 Ms. Siti Hawa

# 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:** 3, Design Patterns and 12 Bit I/O

**Due date:** Wednesday, 30th October 2024, 23:59

**Lecturer:** Ms. Siti Hawa

## Your name: Amani Kamaruddin bin Mikhail Raj Your student ID: J21035623

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:



Problem Set 3

Problem Set 3 - 12-bit I/O

ifstream12.cpp

|  |
| --- |
| #include "ifstream12.h"  #include <cassert>  #include <iostream>  using namespace std;  ifstream12::ifstream12(const char \*aFileName, size\_t aBufferSize) : fBuffer(new byte[aBufferSize]), fBufferSize(aBufferSize),                                                                      fByteCount(0), fByteIndex(0), fBitIndex(-1)  {      if (aFileName)      {          open(aFileName);      }  }  ifstream12::~ifstream12()  {      close(); // Close the file if it is open      delete[] fBuffer;  }  void ifstream12::open(const char \*aFileName)  {      fIStream.open(aFileName, ifstream::binary); // Open file in binarY!!! dik why linux this work      assert(fIStream.is\_open());                 // Success on opening the file      // cout << "File opened: " << aFileName << endl;  }  void ifstream12::close()  {      fIStream.close();      // cout << "File closed." << endl;  }  bool ifstream12::isOpen() const  {      return fIStream.is\_open();  }  bool ifstream12::good() const  {   //good ;-;      return fIStream.good();  }  // Checks if the end of the file and the buffer are both exhausted  bool ifstream12::eof() const  {      //i hate this part of the code please help      bool isExhausted = fIStream.eof() && (fByteCount == 0) && (fBitIndex == 7);      // bool fileStreamEOF = fIStream.eof();          // Check if the file stream has reached EOF      // bool noBytesLeftInBuffer = (fByteCount == 0); // Check if there are no bytes left in the buffer      // bool allBitsRead = (fBitIndex == 7);          // Check if all bits in the current byte have been read      // cout << "Checking EOF - fIStream.eof(): " << fIStream.eof()      //           << ", fByteCount: " << fByteCount      //           << ", fBitIndex: " << fBitIndex      //           << ", eof() result: " << isExhausted << endl;      return isExhausted;      // bool isExhausted = fileStreamEOF && noBytesLeftInBuffer && allBitsRead;  }  void ifstream12::reset()  {      fByteIndex = 0; // Reset byte index to the beginning      fBitIndex = 7;  // Reset bit index to the most significant bit  }  void ifstream12::fetch\_data()  {      if (fByteCount == 0)      {          fIStream.read(reinterpret\_cast<char \*>(fBuffer), fBufferSize);          fByteCount = fIStream.gcount(); // count the byte that actual read          // Force EOF detection if no bytes were read          if (fByteCount == 0 && fIStream.peek() == EOF)          {              fIStream.setstate(ios::eofbit); // Set EOF flag explicitly          }          reset();          // cout << "Fetched " << fByteCount << " bytes into buffer." << endl;      }  }  optional<size\_t> ifstream12::readBit()  {      if (fByteCount == 0)      {          fetch\_data();          if (fByteCount == 0)          {              return nullopt;          }      }      size\_t currentByte = to\_integer<size\_t>(fBuffer[fByteIndex]);      size\_t bitValue = (currentByte & (1 << fBitIndex)) ? 1 : 0;      fBitIndex--;      if (fBitIndex < 0)      {          fBitIndex = 7;          fByteIndex++;          fByteCount--;          if (fByteCount == 0 && !fIStream.eof())          {              fetch\_data();          }      }      // cout << "Read bit: " << bitValue      //           << ", fByteIndex: " << fByteIndex      //           << ", fBitIndex: " << fBitIndex      //           << ", fByteCount: " << fByteCount      //           << endl;      return bitValue;  }  // // Overloaded >> operator  // ifstream12 &ifstream12::operator>>(size\_t &aValue)  // {  //     aValue = 0;  //     for (int i = 0; i < 12; i++)  //     {  //         auto bit = readBit();  //         if (!bit.has\_value())  //         {  //             // cout << "Reached EOF while reading 12 bits." << endl;  //             break;  //         }  //         aValue |= (bit.value() << i);  //     }  //     // cout << "Read 12-bit value: " << aValue << endl;  //     return \*this;  // }  ifstream12 &ifstream12::operator>>(size\_t &aValue)  {      aValue = 0;      // Loop to read 12 bits individually      for (int i = 0; i < 12; i++)      {          // Read the next bit from the stream          optional<size\_t> bit = readBit();          if (bit.has\_value())          {              // Shift the bit to its correct position and add it to aValue              size\_t shiftedBit = bit.value() << i;              aValue |= shiftedBit;          }          else          {              break;          }      }      return \*this;  } |

Output Terminal

|  |  |  |
| --- | --- | --- |
| Window Output terminal   |  | | --- | | g++ .\main.cpp .\ifstream12.cpp .\ofstream12.cpp -o Set3.exe |   Linux Output Terminal   |  | | --- | | g++ .\main.cpp .\ifstream12.cpp .\ofstream12.cpp -o Set3.exe | |