Swinburne University Of Technology

Faculty of Information and Communication Technologies

ASSIGNMENT COVER SHEET

Subject Code:	HIT3303/8303
Subject Title:	Data Structures and Patterns
Assignment number and title:	1, Simple Text Processing
Due date:	March 23, 2011, 10:30 am
Lecturer:	Dr. Markus Lumpe

Your name:____________________

Marker's comments:

Problem	Marks	Obtained
1	69	
Total	69	

Extension certification:

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

Signature of Convener:_________________________

HexDump.h

```
#ifndef HEXDUMP_H_
#define HEXDUMP_H_

#include <fstream>

class HexDump{
private:
		std::ifstream fInput;
		std::ofstream fOutput;

public:
		~HexDump();

		bool open(char* aFileName);
		void close();

		void run();
};

#endif /* HEXDUMP_H_ */
```

HexDump.cpp

```
#include "HexDump.h"
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string.h>

using namespace std;

//Make sure both files are closed when object is deleted
HexDump::~HexDump()
{
	HexDump::close();
}

//Opens the specified input file and corresponding output file - returns false if one of both files can't
//be opened
bool HexDump::open(char* aFileName)
{
	char temp[(strlen(aFileName) + 4)]; //create char array to store file name, with extra 4 characters of length for
						'.txt' appendage
	strcpy(temp, aFileName); //store the input file in a char array, so it can be appended with ".txt'

	fInput.open(aFileName, ios::binary); //open the input file
	fOutput.open(strcat(temp, ".txt"), ios::out); //add .txt to the file, and open it for output

	//check both files opened properly
	if(fInput.is_open() && fOutput.is_open()){
		return true;
	}
	else{
		return false;
	}
}

//Close input and output files
void HexDump::close()
{
	fInput.close();
	fOutput.close();
}

//Run the HexDump application - prints the position of the file pointer (binary), 8 bytes of the file (hex),
//a separator pipe, 8 bytes of the file (hex) and 16 bytes of the file (graphical representation)
void HexDump::run()
{
	int byteCount = 0; //number of bytes count - track position in line; reset each line
	int whiteSpace = 50; //number of whitespace characters needed for an empty line
	string line; //character representation of each line of hex bytes

	while(fInput.good()){
		//test position in line - if at start print the position of the file pointer
		if(byteCount == 0){
			fOutput << hex << uppercase << setfill('0') << setw(8) << fInput.tellg() << ": ";
		}
		//if the next character is the end of the file, print appropriate amount of whitespace, and graphical
		representation of last line
		if(fInput.peek() == -1){
			whiteSpace -= (byteCount * 3);
			if(byteCount >= 8){
				whiteSpace -= 2;
			}
			for(int i = 0; i < whiteSpace; i++){
				fOutput << " ";
			}
			fOutput << line;
		}
		//if next character is not the end of line, get the next character and print as normal
```

```
				else{
					//get next character and add its graphical representation to 'line'
					int nextChar = fInput.get();
					if(isgraph(nextChar)){
						line += (char)nextChar;
					}
					else{
						line += ".";
					}
					//print character in hex, increment byteCount
					fOutput << hex << uppercase << setfill('0') << setw(2) << nextChar << " ";
					++byteCount;
					//test position in line - print separator or graphical representation appropriately
					if(byteCount == 8){
					fOutput << "| ";
					}
					else if (byteCount == 16){
						fOutput << line << "\n";
						line = "";
						byteCount = 0;
					}
				}
			}
		}
```

main.cpp

```
#include "HexDump.h"
#include <iostream>
#include <stdlib.h>

using namespace std;

//Creates a HexDumper, checks if a file to dump was input, and dumps it
int main(int argc, char* argv[])
{
	HexDump* dumper = new HexDump();

	if(argc < 2){
		cout << "No argument given!" << endl;
		exit(1);
	}
	else
	{
		dumper->open(argv[1]);
		dumper->run();
	}

	dumper->close();

	cout << "Dumped successfully." << endl;
	return 0;
}
```