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:_____

Problem Set 1: Simple Text Processing

Build the C++ console application, called `HexDump`, that takes one argument (i.e., a file name), reads the corresponding file in binary mode, and writes a new text file containing the hex dump of the input file. For example, when started in a terminal, then the invocation

```
$> ./HexDump ShiftValue.class
```

reads the file `ShiftValue.class` and produces a hex dump of it that is written to the file `ShiftValue.class.txt`, which should be structured as follows:

```
00000000:  CA FE BA BE 00 00 00 31 | 00 2B 07 00 02 01 00 12  .......1.+......
00000010:  4C 61 6D 62 64 61 46 2F | 53 68 69 66 74 56 61 6C  LambdaF/ShiftVal
00000020:  75 65 07 00 04 01 00 13 | 4C 61 6D 62 64 61 46 2F  ue......LambdaF/
00000030:  42 69 6E 61 72 79 56 61 | 6C 75 65 01 00 06 3C 69  BinaryValue...<i
                                    …

000003C0:  01 00 00 00 39 00 0B 00 | 00 00 20 00 03 00 00 00  ....9..... .....
000003D0:  0E 00 0C 00 0D 00 00 00 | 00 00 0E 00 0E 00 0F 00  ................
000003E0:  01 00 00 00 0E 00 10 00 | 0F 00 02 00 01 00 29 00  ..............).
000003F0:  00 00 02 00 2A                                    ....*
```

Problem Specification

The program HexDump splits the input in units of length 16 bytes. The output format for a unit consists of three parts:

1. Each line of the hex dump begins with the corresponding absolute start address (printed in hexadecimal format) of the unit analyzed.
2. The middle block uses two groups (separated by the character '`|`') of 8 two-digit hexadecimal numbers representing the value at this absolute position in the input file.
3. At the end of each line we print the graphical representation of the unit, that is, a string that represents the printable ASCII character representation of the unit. Byte values that do not possess a graphical representation are substituted by the symbol '`.`' in the output string.

In order to produce the correct output, you will need to use the following I/Omanipulators defined in `iomanip` of the C++ I/O library:

- `endl` – generate platform specific end-of-line,
- `hex` – force integers to be printed in hexadecimal format,
- `setw(n)` – sets the precision of a printed integer,
- `setfill(c)` – sets the fill character for `0`s left of an integer, and
- `uppercase` – print all character in upper case format.

In addition, you may need to study the manual pages (or any appropriate C++ documentation) to learn more about the library classes `ifstream`, `ofstream`, and `string`. The solution requires a proper use of these library classes as well as a good understanding of the stream methods `open()`, `fail()`, `close()`, `read()`, and `gcount()` and the string method `c_str()`.

The application `HexDump` uses file stream objects to read and write files. For this reason, you need to include `iostream`, `fstream`, and `iomanip` where appropriate into the current compilation unit. These header files provide the required abstractions to work with files. You can obtain the output file name by appending the string "`.txt`" to the input file name. Please note that the open method for streams requires a C-string (i.e., a value of type `char*`). The C++ type string offers the method `c_str()` that you should use for this purpose. When working with files remember to define appropriate guards to check for any occurrences of I/O errors, as this will prevent the application `HexDump` from working correctly.

The application should consist of two parts: a class `HexDump` that implements the desired functionality and a main function that performs the necessary argument checks, instantiates an object of class `HexDump`, and calls the appropriate methods in the right order. The specification of class `HexDump` is shown below:

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

public:
    ~HexDump();

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

    void run();
};
```

There may also be the need to use type casts in order to adjust the actual type of an expression to the required type. A typecast is written `(typename)expression`. For example, if you wish to output a character (i.e., a variable of type `char`), say `current_char`, to an output stream, say `ostream`, then you would normally write `ostream << current_char`. Unfortunately, the selected `<<` operator in this case will render the value of `current_char` to its printable representation, which may not be desired. To print the raw value (i.e., an integer representation), you must apply a cast to `int` first. More precisely, when you write `ostream << (int)current_char`, then the output is the integer representation (in hexadecimal if you have used the correct manipulator). To illustrate the different behavior, consider the following code fragment using cout – the standard output stream:

```
char current_character = 'A';

cout << current_character << endl;

cout << (int) current_character << endl;
```

The two output statements produce the following result:

```
A

65
```

So, in the first case we obtain the graphical representation of `current_character`, whereas in the second case the integer representation is generated. The integer

representation uses the default manipulator `dec`. To render it to hexadecimal, you need to use the manipulator `hex`. Please note the effect of a manipulator remains intact until you apply another manipulator.

The final program may not require more than 100 lines of code including comments and very spacious formatting.

In order to achieve full marks, you must correctly implement the functionality including a proper handling of the last line. The separator '|' must occur if there are at least 8 characters and the printable representation must be properly aligned.

Submission deadline: Wednesday, March 23, 2011, 10:30 a.m.

Submission procedure: on paper.