Swinburne University of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code: SWE20004
Subject Title: Technical Software Development
Assignment number and title: 1, Simple Data Processing
Due date: April 5, 2018, 08:30
Submission procedure: on paper, in class
Lecturer: Dr. Markus Lumpe

Your name: _____ **Your Student ID:** _____

Check Tutorial	Wed 14:30	Thurs 12:30	Thurs 14:30	Thurs 16:30	Fri 08:30	Fri 10:30	Fri 12:30	Fri 14:30	Fri 15:30

Marker's comments:

Problem	Marks	Obtained
Analysis	10	
Test	10	
Design	10	
Implementation	57	
Total	87	

Extension certification:

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

Signature of Convener:_____

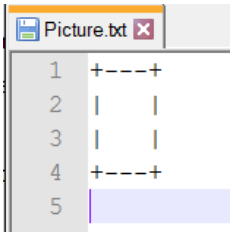
Problem Set 1: Simple Data Processing

Build the C++ Win32 console application, called `DataDecoder`, that takes two command line arguments: a name of a data input text file and a name of a data output file.

The idea of `DataDecoder` is to convert a series of 32-bit integer values into a sequence of printable 8-bit character values. Mapping the 32-bit integers to printable 8-bit character values should result in a clear ASCII art image. ASCII art is a graphic design technique that uses computers for presentation and consists of pictures pieced together from the 95 printable (from a total of 128) characters defined by the ASCII Standard. For example, the input

```
757935403 544999979 175906848 538976380
757795452 170601773
```

represents the ASCII art picture (shown below in Notepad++)



a simple rectangular box. In other words, the six 32-bit integers represent 24 ASCII characters that make up the rectangular box (using the UNIX-style newline character `'\n'`).

A 32-bit integer can encode four printable 8-bit characters. These characters are arranged in the integer value from right to left (i.e., the least significant byte in the 32-bit integer contains the first character). For example, the value `757935403` corresponds to the characters `'-'`, `'-'`, `'-'`, `'+'`. It may help to view `757935403` in hexadecimal representation: `0x2D2D2D2B`. When inspected in hexadecimal representation, the value `757935403` clearly contains four elements: `2D`, `2D`, `2D`, `2B` that are the ASCII codes for the characters `'-'`, `'-'`, `'-'`, `'+'`.

The input text file contains a sequence of 32-bit integers in decimal format separated by spaces or newlines. Spaces and newlines are white-space characters whose sole purpose is to divide a sequence of numbers into 32-bit integers. You have to use formatted input to read those 32-bit integers. For example, the sequence `538976288 538976288 538976288 538976288` stands for four 32-bit integers. Please note that the input data is well formed. Your program should detect EOF after reading the last 32-bit integer. The file `InputData.txt` ends with a newline character. Reading it triggers EOF and *input is not good*. You can use this condition to stop the conversion from 32-bit integers to ASCII art picture.

A printable 8-bit character value requires 1 byte (i.e., 2^8 possible values). Using the fact that $2^8 = 256$, we can split a 32-bit integer into four pieces. We can use the modulo operator and integer division for this purpose. For example, computing `757935403 % 256` gives `43` (or `0x2B` in hexadecimal), which corresponds to character `'+'` in the ASCII table. In other words, we can use modulo to extract a character from right to left. To access the next character, we need to "shift" the 32-bit integer value by 8 bits to fetch the next character. This is a simple integer division by 256 (i.e., 2^8). For each 32-bit integer, we need to repeat these two steps (modulo and integer division) four times. This process requires at least two variables: an `int`

variable to store the 32-bit integer (including the result of integer division) and a `char` variable to host the decoded character (result of modulo).

An ASCII art picture usually contains the newline character `'\n'`. The input data for this assignment uses the value `0x0A` (decimal 10) to represent the newline character. This is a UNIX/Mac convention. On Windows computers, this does not work. To make `DataDecoder` work properly on both Windows and UNIX/Mac computers, we cannot just simply send `0x0A` to the output to mean the newline character. We need to add a test: if the value of the decoded character is equal to `0x0A`, then we output `'\n'` (or `endl`). Otherwise, we output just the decoded character.

Tasks

Follow the design and development steps:

1. Perform a problem analysis and identify the functional and non-functional requirements to build `DataDecoder`. (on paper)
2. Create and study a test scenario using the sample data (rectangular box). In the test scenario, you should experiment and study how a 32-bit integer can be split into four ASCII characters. Please remember that ASCII is a coding standard for characters. It states, for example, that the value 64 maps to character 'A'. (on paper)
3. Outline, identify, and design the main building blocks of the application. You should develop the algorithm in pseudo-code, if possible. (on paper)
4. Implement the solution and test it with the file `InputData.txt`. (printed program code and result from test run)

`DataDecoder` is a Win32 console application that reads an input text file and produces another text file as output. The names of these files are given as command line arguments. The solution requires simple control structures and using integers as characters.

Your application also needs to produce a "production maker". This information must be written to the output file (and be part of the program). For example, `DataDecoder` has to convert the input

```
757935403 544999979 175906848 538976380

757795452 170601773
```

into the ASCII art picture

```
+---+
|   |
|   |
+---+


Prepared by StudentName (StudentID)
```

where `StudentName` is you name and `StudentID` is your student id.

The implementation requires less than 100 lines of code. The file `InputData.txt` encodes an ASCII art picture of a well-known movie character.

Submission deadline: Thursday, April 5, 2018, 08:30.

Submission procedure: on paper (printed solution and cover sheet).

Sample Input (InputData.txt)

```
538976288 538976288 538976288 538976288 538976288 1600085855
537534559 538976288 538976288 538976288 538976288 539438380
538976288 543108704 538976266 538976288 538976288 538976288
538978092 538976288 538976288 539910176 538976288 740302880
169881133 538976288 538976288 538976288 539438112 538976288
538976288 538976288 542908448 538976288 740892704 169892910
740302880 538976302 538976288 538976303 538976288 538976288
538976288 1545609248 538976288 539504416 540761120 790634506
542923815 538976288 539914028 538976288 740302880 1869573999
538979951 538991648 538978092 690433376 538970656 689973545
1679830624 1495420984 538979896 942415904 572662352 945365538
1612718126 773857319 656551213 673188384 539438432 1499471904
1612718119 538982489 1348739104 538976288 1612718112 538976295
539959328 1612718090 757935405 538978350 543121184 538976352
539438112 757935404 538976302 538976288 537534504 538976288
538976288 757935145 539910190 539500576 757873440 539897133
538976288 689971232 538970656 538976288 539959328 744443740
539566119 542908448 1599036448 538978092 538976288 545005600
538976266 538976288 1612718139 740306221 538976295 538976288
757935456 538976295 538976288 169901088 538976288 2082480160
538976288 538978093 538976288 1612718112 538976302 538976288
538976288 537534588 538976288 540761888 740302880 538976288
538976288 538976288 538976297 538976288 975183904 538970656
657350432 538976380 778051616 543108666 774643744 539114042
539897440 538976288 545005600 657272074 2082480160 773857312
538976295 572661792 538976290 538976288 543170592 538976288
778075168 538970656 538976288 975183996 538976315 975183904
975183904 538976288 539909920 538976288 1618747424 758013998
774712615 538970656 538976288 656416892 538979872 975183904
975183904 741236575 539982887 538976288 545005600 169892896
538976288 543170592 538976288 774712668 758013791 2086632487
791510111 538976288 538976288 538976303 537534505 538976288
542908448 538976288 538992476 757948448 1600070238 538978092
538976288 539762720 2082480160 538970656 538976288 538983200
538976352 757948448 757935454 538976295 538976288 740302880
538976295 545005600 538976266 538976288 538991648 538976352
538976288 538976288 538976288 538976288 538980128 538976288
```

```
537534511 538976288 538976288 538976348 538976352 1595957024
538976288 538976288 539959328 538976266 538976288 1545609248
538976288 538976288 1612718112 538976288 790634528 538970656
538976288 538976288 538991648 538976288 538976288 538978080
657203232 538970656 538976288 538976288 778051616 538976288
740302880 1595940896 169879340 538976288 538976288 538976288
761274400 1600085806 757935406 538976295
```