Course: CpSc 212: Algorithms and Data Structures Spring 2014
Lab 04: Conversion of Bit Map Formats using Divide and Conquer

In this lab, you will be using divide and conquer and recursion to convert a bit map between its two representations. A bit map is a matrix (double dimensional array) in which each element is either a 0 or 1. You can think of bit map as a boolean matrix where 0 represents *false* and 1 represents *true*. These are used a lot in practise. For instance, in computer graphics, a value of 0 may represent a black pixel, while 1 may represent a white pixel.

1 The Two Formats

A bit map can be represented as nothing but a double dimensional integer array with m rows and n columns. We will call this representation as "format B" due to lack of a better word. An example of a bit map in format B is given below.

0	0	0	0	1	1	1	1
0	0	0	0	1	0	0	1
0	0	0	0	1	0	0	1
0	0	0	0	1	1	1	1
1	1	1	1	0	0	1	0
1	1	1	1	0	0	1	0
1	1	1	1	1	1	0	1
1	1	1	1	1	1	0	1

We would like to apply divide and conquer to change the bit map into another representation called "format D", again due to lack of a better term. We use the following algorithm to convert from format B to format D.

1. If all entries in format B are 0, then output a 0.
2. Else if all entries in format B are 1, then output a 1.
3. Else:
   (a) Output a D (which means divide)
   (b) Divide the matrix in the middle. That is, divide the matrix into four quadrants, each with $m/2$ rows and $n/2$ columns. The four quadrants are respectively top-left, top-right, bottom-left and bottom-right. Technically, the top-left, top-right, bottom-left and bottom-right quadrants have sizes $\lceil m/2 \rceil \cdot \lceil n/2 \rceil$, $\lceil m/2 \rceil \cdot \lfloor n/2 \rfloor$, $\lfloor m/2 \rfloor \cdot \lceil n/2 \rceil$ and $\lfloor m/2 \rfloor \cdot \lfloor n/2 \rfloor$ respectively.
   (c) Recursively compute format D representation of the top-left quadrant.
   (d) Recursively compute format D representation of the top-right quadrant.
   (e) Recursively compute format D representation of the bottom-left quadrant.

(f) Recursively compute format D representation of the bottom-right quadrant.

The format D representation of the above matrix is:

```
D0DD1110D1101D1011D01111D0D10101D0101
```

The conversion from format D to B is also straightforward. If a single 0, or 1 is read, fill the matrix with 0 or 1 respectively. Otherwise, recursively fill the top-left, top-right, bottom-left and bottom-right quadrants in order.

Note that during recursion, we always process the quadrants in the following order: top-left, top-right, bottom-left and bottom-right. Your task in this lab is to convert a bit map representation in format B to format D and vice versa.

1.1 Helper Code

Please first download the *lab04.tar.gz* file from the following webpage:

```
http://people.cs.clemson.edu/~rmohan/course/cpsc2120-sp14/
```

Next, please extract the files using the following command.

```
tar -zxvf lab04.tar.gz
```

This command will extract all the files in the directory named *lab04*. Please change your directory using the command:

```
cd lab04
```

You are to only write two functions in *main.cpp*. Please complete the functions *convertBtoD* and *convertDtoB*. We have already provided the input and output for you. But make sure you understand the rest of the code. You should not modify the rest of the code.

1.2 Input and Output

The first line of your input contains a character that defines the input representation followed by the dimensions of the matrix (first the number of rows m and then the number of columns n, $1 \leq m, n \leq 200$). Each of these are separated by a space. Allowable formats are B and D. If the format is 'B', then the next m lines contain exactly n numbers separated by a space. If the format is 'D', then the next line gives the entire format D representation of the bit map.

If the input format is 'B', then output the entire format 'D' representation on a single line. If the format is 'D', then output m lines, each containing n numbers that give the corresponding format B representation. Sample input and output is shown in the following table.

Sample I/O

Test case#	Input	Output
1.	B 8 8 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1	D0DD1110D1101D1011D01111D0D10101D0101

Test case#	Input	Output

2.	D 8 8
	D0DD1110D1101D1011D01111D0D10101D0101
		0 0 0 0 1 1 1 1
		0 0 0 0 1 0 0 1
		0 0 0 0 1 0 0 1
		0 0 0 0 1 1 1 1
		1 1 1 1 0 0 1 0
		1 1 1 1 0 0 1 0
		1 1 1 1 1 1 0 1
		1 1 1 1 1 1 0 1

1.3 Compiling and Testing

Please compile your code using the following command:

```
g++ main.cpp -o main
```

We have provided two input files *input1* and *input2* that contain two representations of a bit map. Please redirect the input files while executing your code. That is execute using the command:

```
./main < input1
```

Please thoroughly test your program and make sure it works for all cases. In particular, please make sure your program works when the number of rows is not equal to the number of columns and the dimensions of the matrix are $1 \cdot 1$ and $200 \cdot 200$. Please always enter your input in a file and redirect while executing your program.

2 Submission

Please submit your *main.cpp* file through handin.cs.clemson.edu. The due date for this assignment is Friday February 14th at 11:59pm.

3 Grading

Ten points is awarded for each conversion. The total possible grades for this assignment is 20. No points will be awarded to a program that does not compile. You will recieve 1 point for successful compilation and 1 point for successful execution (no seg faults and infinite loops). 3 points is awarded for correctness and efficiency each and 2 points awarded for good code: readability, comments, simplicity and elegance, etc.