Due Date: February 26th, 2010

CIS 296	Assignment #2	Winter 2010

Program 1: Cryptic Condiments

Ginger Root and Tabasco Sauce have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message down the columns, padding with extra random letters so as to make a rectangular array of letters. A message may consist of upper and lower case letters, numbers, spaces and punctuation. For example, if the message is "There's no place like home on a snowy night" and there are five columns, Ginger Root would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Root included only letters and writes them all in lower case. This may not be the case with other input. In this example, Root used the character 'x' to pad the message out to make a rectangle, although he could have used any letter. Ginger then sends the message to Tabasco Sauce by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover the original message for Tabasco Sauce (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2 . . . 20 indicating the number of columns used. The next line is a string of up to 200 lower or upper case letters, digits, spaces and punctuation. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output in a GUI Dialog window, giving the original plaintext message, with no spaces.

Sample Input

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Sample Output

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

Due Date: February 26th, 2010

Grading for Crypto Condiments:

- Use `StringBuilder` class. i.e., you can use it to reverse some strings
- You may not use the `String` class in your program. (The `String` in `main(String[] args)` does not count)
- Your program must read the data from a file called "input.txt"
- Your program must write the output to a file called "output.txt"
- Program must also display the output using GUI elements.
  For example, for each test case you can show a Dialog window with a line of output. Upon clicking Okay, the next line of output will be shown in a dialog. See 1.10 (p. 19) on how to do this. If you are GUI-proficient, feel free to use any other way of GUI output
- All input will be within specifications
- Use the provided sample input and output files for testing (download them from VLT)
- Final program will be run on larger input file, where the input will be within specifications, but unknown to you.
- I will be using contest-style verification for the output file. Your output file must exactly match the judge's output file, character by character. Use the sample output file on VLT to compare. This means that user prompts, such as "Please enter string" may mark down your score, if they end up in the output file.
- Upload your program's Java file to VLT's Assignment 2 folder. Only the Java file will be used for grading

Hints for Better Programming

- Work the problem logic out on paper
- Break up complex ideas into smaller and smaller parts until they become trivial steps
- Translate those trivial steps into code
- Use the book , the notes and the web to look up Java syntax
- Write and test small portions of code at a time