Co(w)mpression Challenge

How much of a USACO problem statement is actually important for solving a problem? In this challenge, you will work in teams to build a program capable of compressing problem statements, with a goal that the can subsequently be reconstructed well enough to solve the original problem!

1 Program Structure

Your team will write a compression program that reads input from *statement.in*, a text file containing the problem description of an algorithmic programming task written in English in the style of a USACO problem. Just as with any other USACO problem, you can expect to find a textual narrative for the problem followed by sections "INPUT FORMAT", "OUTPUT FORMAT", "SAMPLE INPUT", and "SAMPLE OUTPUT", and then possibly an explanation of the sample input / output. All math will be encoded in plain text, so for example you might see a subscript written as `A_7` and a superscript written as `x^2`. All mathematical expressions will be delimited with dollar signs[1].

Your program will also accept a single parameter on the command line, an integer c in the range $0 \ldots 99$, indicating the amount of compression to apply. If the input file `statement.in` contains b bytes, your compressed output may have length at most $\lceil (100 - c)b/100 \rceil$ bytes. Hence, $c = 0$ indicates no compression, and $c = 99$ indicates that the result should be only 1% of the size of the original! To get the size of a file and then read the entire contents of the file into a buffer, you can use the following C code:

```
unsigned char *buffer;
FILE *fp = fopen ("statement.in", "r");
fseek (fp, 0, SEEK_END);
int filesize_in_bytes = ftell(fp);
fseek (fp, 0, SEEK_SET);
buffer = (unsigned char *)malloc (filesize_in_bytes);
fread (buffer, filesize_in_bytes, 1, fp);
fclose (fp);
```

As output, your program should write a file `statement.out` containing the result of compressing the input file. The length of `statement.out` in bytes must satisfy the length constraint above or your program will be disqualified.

[1] This is just like using LaTex, if you are familiar with this method of markup for mathematics. However, problem statements won't include LaTex commands like `\leq` or `\frac`.

2 Due Date

Please upload your final program to your individual shared Dropbox folder by no later than Thursday May 29 at 2pm EST. Please have only one team member upload code to their shared Dropbox folder, but include in a comment in the code who are the members of the team.

3 Reconstruction / Competition

At the end of the week (on Saturday afternoon), campers will compete in a fun challenge contest where you will be given problem statements that have been compressed using your code to various levels. As such, you may want to collaborate with your partner in advance to also write tools that support decompression of your compressed problem statements, making it easy to recover as much information as possible. During this contest, you will be allowed to communicate with your teammate.