S-Q 1997 - Assignment 6
Makeup Assignment

Due 08/07/97 at 6:00pm

Last Date Accepted: 08/11/97 at 6:00pm

Note: this is a makeup for one of the first three assignments. If you hand in a solution for any part of this assignment, your score for this assignment will replace your lowest score on any of the first three assignments. (Note that the graduate problem is treated differently, see below.)

1 Getting Prepared

1. Copy over the files for this assignment. Copy all of the files from `~libsq/assts/asst6` into your directory `~/Q/asst6` (which should have been created when you ran `q-setup` for assignment 1).

2 Exercises (100 points)

All work should be done in directory `~/Q/asst6`. The `Makefile` provided includes rules to build all the programs in this assignment; you shouldn't need to change it at all.

You may write your answers to the written questions in whatever manner you prefer– longhand, with UNIX tools, or via a word processor. Make sure to hand in a copy of whatever is appropriate given your choice.

1. (35 points) Weiss, 2.11. Specify your algorithm completely (in C, if you wish), and be sure to analyze the running time of your algorithm.

   There is an obvious O(N) solution– just work your way down the array testing each element in turn. Your algorithm should be much faster. Hint– this problem would be much harder if the array was not a sorted array of integers!

2. (30 points) What is the probability that a randomly selected 3-character string will appear as a word in the UNIX spelling dictionary?

   Complete the program named `monte.c`, to compute this probability with a confidence level of 99% and a margin of error of 1%.

   For the sake of this problem, only consider the lower-case letters 'a' through 'z'.

   A modified version of the hash tables from assignment 4 (along with the code to load the dictionary into a hash table) has been provided. Consult `monte.c` for more information. You may also find the pseudo-random functions in `random.c` to be useful.

   Note that this exact problem is relatively silly; it would be less work to simply count the exact number of each of the kinds of words we are looking for by scanning the dictionary once than it would be to perform this experiment. However, if we were sampling from something that we *couldn't* scan entirely, this method would still work well.

3. (35 points)

   (a) Edit file `rle.c` .c to complete the `rle` (Run Length Encode) program, which implements run length encoding according to the following rules:

      - Any sequence of between 2 and 255 identical bytes is encoded by a two-byte code that consists of the count (as the first byte in the encoding) and the value of the repeated byte (as the second byte in the encoding).
      - Any sequence of bytes that do not contain repeated characters is represented by a byte with a value of 1 followed by the sequence of bytes, terminated with another 1. If a 1 appears as part of the sequence, it is escaped with a 1.

      For example, the string "AAAAAABCCCC" would be represented by the following 7 bytes: 6, 'A', 1, 'B', 1, 4, 'C'.

      Your `rle` program should read its input from standard input (stdin), and write its output to standard output (stdout).

      The UNIX utility od may be very useful, because it allows you to look at the values represented by bytes as numbers and/or ASCII characters.

      A run-length decoder (`sol_rld`) and example solution encoder (`sol_rle`) are available to help you test your `rle`.

   (b) Analyze this encoding and your implementation. Answer the following questions:

      - What is the big-Oh of your encoding program, in terms of the size of the input?
      - What is the maximum compression ratio (the ratio of input to output sizes) that this coding can achieve? What is the minimum compression ratio? Describe inputs that achieve these limits.

3 Graduate Problem (25 points)

Weiss, problem 7.34.

Note: this is a makeup for the graduate problem on assignment 3 only. If you hand in a solution for this problem, your score for this problem will replace your score on the graduate problem from assignment 3.

4 Submit Your Work

1. Use the `submit` program to hand in your work. From your directory `~/Q`, run the `submit` program. The name of the course is **cscisq**, and the name of the project is **asst6**, and the name of the file to submit is also **asst6**. (See the UNIX tutorial for more info about `submit`.)

2. Print out a copy of all of the text or code files you created or modified as part of this assignment, and hand them in at the next class or section. If you wrote out the answers to your questions longhand or using a word processor, include printouts of the final document.