Assignment 2a: sbsrm

Due: Wednesday February 6, 11:59 p.m.

Your task is to implement the remove command for the simple backup system you implemented in Assignment 1. This command will be implemented as a Bourne shell program called sbsrm.

The sbsrm program takes one or more arguments. An argument can be an absolute or relative path to a file or directory. The program performs three operations on each argument: removing the file or directory from the appropriate .sbs file, removing the file or directory from the working directory, and removing the file or directory from the appropriate location in the repository. If the file or directory is not in a .sbs file, then an error message should be printed, and no files or directories should be removed. If the argument is a directory, sbsrm should remove the directory in the repository or the working directory only if that directory is empty (see rmdir). You do not need to handle file or directory names with spaces in them.

Two C programs are given to you (see the web page): dbfind and dbremove. You may not change these two programs, and you should not submit them. The Makefile includes an install rule which copies the executables dbfind, dbremove, and sbsrm to ~/bin. You should create a directory called bin in your home directory and add it to the your PATH variable (see the web page). Ensuring that the programs are in ~/bin and ~/bin is in your path allows you to execute sbsrm from any directory. Please ensure that only permissions for everyone else are turned off for ~/bin (man chmod). Do not put paths to any of these programs in the sbsrm program.

Two commands that you will find indispensable are dirname and basename. See their man pages to learn how they work. A final tip is that you do not necessarily need to use your sbs program from Assignment 1 to test this program. All you need to do is create the appropriate files.

What to submit

For this assignment you will submit one file called sbsrm containing a Bourne shell program. A full solution was implemented in approximately 80 lines including comments and white space. Excessively long or poorly organized solutions will be penalized.

You will also submit a plain text file called testcases containing a list of the test cases and expected results used to test this program. A good set of test cases thoroughly tests the program in as few cases as possible. Marks will be awarded for completeness, conciseness, clarity, and writing quality. Please ensure that lines of the file do not exceed 80 characters.