CSC209H Worksheet: Shell Programming: Part 2

1. Before you start this exercise, create a new directory and `cd` into it. Now complete the following table by typing each expression into the bash shell. If the command produces an error, give the error message. Otherwise, show the result printed to standard output.

Expression	Error? (Y/N)	`stdout` or Error message
`song="national anthem"` `echo song`		
`echo O Canada, our home and  > $song` `ls -l`		
`echo "O Canada, our home and > $song"` `ls -l`		
`echo O Canada, our home and > "$song"` `ls -l`		
`cat $song`		
`echo Who has seen the wind > story` `ls | wc`		

`for i in *; do`
`  echo $i is a file`
`done`		

2. Recall the program `fibonacci` that you wrote for the dynamic memory lab. It took a single integer command-line argument and then wrote a message to `stdout`. Write a shell program that will take multiple arguments (each of which are integers) and call `fibonacci` on each argument. The original fibonacci starter code doesn't print a newline character at the end of the output. Change the main function of `fibonacci.c` to add a newline so that each run of your program appears on its own line.

3. Suppose you have a program `floop` that takes two command-line arguments: the first is an integer and the second is a filename. Write a script that will itself take two command-line arguments. The first will be an integer `upper` and the second will be the filename. Your script should repeatedly call `floop` using that filename and every integer from 1 to `upper`. Whenever `floop` returns a non-zero value, you should report that that integer/filename combination is "floopy". You should discard the standard output from `floop`. We have provided a `floop` executable at `/u/csc209h/winter/pub/shell-prog/floop`.

4. In lab, you wrote the program `time_reads`, which takes arguments representing a number of seconds and the name of a test file. Write a shell script that takes a number of trials `n` and a filename. Your script should run `n` trials of your `time_reads` program, each time for 2 seconds, and print the average number of reads over these `n` trials. Hint: Start by making sure you can run `time_reads` once and extract the number of reads from the output and store this in a variable. Use the command `cut`.