CSCI-14 Lab 10, due 4/26/18

Please work in groups of 2. Consider a guess-a-number game and an optimal algorithm for a player to play the game.

The guess-a-number game works like this: the computer selects a random number in the range of 1 to 100, inclusive, and the user tries to guess the number in no more than 8 tries. The computer's responses are "Congratulations, you got my number in only N tries", "Your guess is too big", "Your guess is too small" and "Sorry, you guessed eight times without getting the number, my number was N".

Given that information, what is the best algorithm for the player to get the number? Is it possible for the player to always win the game? It is, and your job is to work out both algorithms.

First, design the algorithm for the computer to choose a random number and then give you at most eight guesses to get it. This will become a program "chooser.cpp".

Then design an algorithm for the computer to guess a number you have chosen, limiting itself to eight guesses and having you respond in a way that indicates too big, too small or correct. This will become a program "guesser.cpp".

After designing the algorithms, show me your top-down designs in lab. After I O.K. them, you may write the two programs. Then, show me the programs playing against each other before you email them to me. Send me the source files and a couple of test runs.

If you `#include <cstdlib>`, you will get access to two useful functions prototyped here:
`int rand();` and
`void srand( int seed );`

Each call to `rand()` returns a different pseudo-random integer in the range 0…RAND_MAX. RAND_MAX is a system-specific value for the largest integer `rand()` can return. `srand()` takes a seed, used to select a new pseudo-random number in the next call to `rand()`. You only seed with `srand()` ONCE. Then you may use `rand()` as often as needed.

Another useful header is `<ctime>`. `ctime` gives you access to a function
`int time( struct tm *p );`
Don't worry about what a `struct tm` is, or what a pointer to one is. A call like
`int t = time( NULL );`
will give you the current system time in seconds since midnight, January 1, 1970. NULL is just a pointer to a memory location that cannot be accessed, so time() will ignore it. The value from time() can be passed into a call to srand() to set up different sequences from the random number generator on each run of the program.

rand() % 100 gives a random number in the range 0…99. How can you get a number in the range 1…100?