overview

- This is the project for unit V of cis 1.5. This project covers searching and (optionally) sorting. It also serves as a review of arrays and functions.
- The project is worth 10% of your term grade. It will be marked out of **10 points**.
  Note that there is an **extra credit** option worth **5 additional points**.
- The project is due via email on **Monday May 11**.
- Email your C++ file (**lucky.cpp**, as described below) to: `sklar@sci.brooklyn.cuny.edu`.

project description

This project simulates a guessing game.

1. Create a file called **lucky.cpp**.
2. Define a constant called `SIZE` and set it equal to 5. *(0.2 points)*
3. Inside the `main()`, declare an array called `game` that contains `SIZE` integers. *(0.3 points)*
4. Create a `void` function called `initGame()` that takes two arguments: an integer array and a single integer indicating the length of the array. The function should initialize each entry in the array to a value between 1 and 1000. *(1 point)*
5. Create a `void` function called `printGame()` that takes two arguments: an integer array and a single integer indicating the length of the array. The function should print out the contents of the array. *(1 point)*
6. Create an integer function called `guess()` that asks the user to enter a value between 1 and 1000, reads in the value and returns the user's value. The function should validate the user's input (i.e., make sure that s/he entered a value between 1 and 1000). The function should loop until the user has entered a valid value. *(2 points)*
7. Create a boolean function called `found()` that takes three arguments: the integer array of numbers, the size of the integer array, an integer value to search for. The function should search for the value (third argument) in the array (first argument) and return `true` if the value is found in the array or `false` if the value is not found in the array.
   Use a linear (sequential) search on the unsorted array. *(2 points)*
8. Finish the `main()` so that: *(2.5 points)*
   - First, it calls `initGame()` to initialize the values in the `game` array declared above.
   - Then, it calls `printGame()` to print out the contents of the `game` array.
   - Note that you will later comment out the call to `printGame()`, but for now we leave it in so that you can make sure that your code works properly.
   - Third, your program should call the `guess()` function to ask the user to guess a value.
   - Fourth, call `found()` which should look to see if the value that the user entered is in the `game` array.
   - Fifth, the program should print a message to the user letting her know if her value was found in the game (i.e., in the `game` array).

- Put these steps (third, fourth and fifth) inside a loop, asking the user if she wants to play again after she has guessed once.
- Test your code and make sure that it works.
- If it works, then comment out the call to `printGame()` and change the value of `SIZE` from 5 to 100—and then your guessing games is complete!

9. Play your game lots of times. On average, how many guesses does it take to find a number in the array? Include your answer in your email when you submit your code. *(1 point)*

EXTRA CREDIT (5 points)

Modify the program to include a function that **sorts** the array after it has been initialized, using one of the sorting algorithms discussed in this unit; and then uses **binary search** (*hint:* implement it in the `found()` function) to try and locate the user's guess. *(2 points for sort; 3 points for binary search)*

submission instructions

- You will be submitting ONE file: **lucky.cpp**
- Make sure that you have a COMMENT at the top of the file that contains the name of the file, YOUR NAME, "CIS 1.5 PROJECT 5" and the submission date (May 11, 2009).
- The SUBJECT LINE of your email should say: CIS 1.5 PROJECT 5
- The BODY of your email should contain your name.