Mike’s teacher told his class that a flipped coin lands on “heads” half the time and “tails” the other half. “That means that if I flip a coin 100 times, it should land on heads exactly fifty times!” said Mike. Mike’s teacher knows that the law of probability does not quite work that way. To demonstrate this principle, she decides to write a program.

**Program**

Write a function to simulate a coin flip, returning true if the coin lands on “heads” and false if it lands on “tails.” Next, prompt the user for how many trials will be needed for the experiment. Finally, display how many “heads” and “tails” were recorded in the experiment.

**Example**

User input is **Bold**.

<table>
<thead>
<tr>
<th>How many coin flips for this experiment: 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>There were 49 heads.</td>
</tr>
<tr>
<td>There were 51 tails.</td>
</tr>
</tbody>
</table>

**Assignment**

Please:

- Start from the standard template we use for homework assignments:
  
  /home/cs124/templateTest2.cpp

- Make sure your professor’s name is in the program header.
- Run test bed with
  
  testBed cs124/practice21 test2.cpp
- Run style checker

Note that the following code might come in handy:

```c++
#include <stdlib.h>    // needed for the rand(), srand()
#include <ctime>       // needed for the time function
int main(int argc, char **argv)
{
    // this code is necessary to set up the random number generator. If
    // your program uses a random number generator, you will need this
    // code. Otherwise, you can safely delete it. Note: this must go in main()
    srand(argc == 1 ? time(NULL) : (int)argv[1][1]);

    // this code will actually generate a random number between 0 and 999
    cout << rand() % 1000 << endl;
}
```
## Grading

<table>
<thead>
<tr>
<th></th>
<th>Exceptional 100%</th>
<th>Good 90%</th>
<th>Acceptable 70%</th>
<th>Developing 50%</th>
<th>Missing 0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions 10%</td>
<td>The expression for the equation is elegant and easy to verify</td>
<td>The expression correctly computes the equation</td>
<td>One bug exists</td>
<td>Two or more bugs exist</td>
<td>The expression is missing</td>
</tr>
<tr>
<td>Modularization 20%</td>
<td>Functional cohesion and loose coupling is used throughout</td>
<td>Zero syntax errors in the use of functions, but room exits for improvements in modularization</td>
<td>Data incorrectly passed between functions</td>
<td>At least one bug in the way a function is defined or called</td>
<td>All the code exists in one function</td>
</tr>
<tr>
<td>Loop 40%</td>
<td>The loop is both elegant and efficient</td>
<td>The loop is syntactically correct and used correctly</td>
<td>The loop is syntactically correct or is used correctly</td>
<td>Elements of the solution are present</td>
<td>No attempt was made to use a loop</td>
</tr>
<tr>
<td>Output 20%</td>
<td>Zero test bed errors</td>
<td>Looks the same on screen, but minor test bed errors</td>
<td>One major test bed error</td>
<td>The program compiles and elements of the solution exist</td>
<td>Program output does not resemble the problem or fails to compile</td>
</tr>
<tr>
<td>Style 10%</td>
<td>Well commented, meaningful variable names, effective use of blank lines</td>
<td>Zero style checker errors</td>
<td>One or two minor style checker errors</td>
<td>Code is readable, but serious style infractions</td>
<td>No evidence of the principles of elements of style in the program</td>
</tr>
</tbody>
</table>

The solution is available at

`/home/cs124/tests/practice21.cpp`