Exceptions

Modify and improve the random number generator from exercise 5.

This is what the class from exercise 5 should do:

> Constructor takes two parameters that determine the range of numbers that the generator produces. The parameters are upper and lower limit of the range. Both values are included in the range. For example: lower = 2, upper = 6 can produce following random numbers: 2, 3, 4, 5, 6 (in random order).

The generator in the exercise 5 gets stuck in an infinite loop if all numbers in the range have already been used. Improve the generator so that it throws **`runtime_error`** with explanation *"Unable to produce unique random number"* if all numbers in the range have already been used.

Implement function **`test_generator`** that generates requested number of unique random numbers. Function catches exceptions and prints a message if an exception occurs. Finally, the program prints the generated random numbers. The following example assumes that the class is called UniqueRng. Change the class name to match your random number class name.

```
void test_generator(UniqueRng ur, int count)

int main(void) {
    UniqueRng randGen(5, 13);

    test_generator(randGen, 6);
    test_generator(randGen, 9);
    test_generator(randGen, 13); // this will cause exception

    UniqueRng randGen2(1, 35);
    test_generator(randGen2, 7);
    test_generator(randGen2, 7);
    test_generator(randGen2, 7);
    test_generator(randGen2, 70); // this will cause exception
}
```

Example output (partial) on the following page.

Example output:

```
Generating numbers:
7
10
11
6
13
9
End of generator
Generating numbers:
9
6
10
8
13
12
5
7
11
End of generator
Generating numbers:
Exception: Unable to produce unique random number. Tried to generate 13 random
numbers. Got only 9
8
7
9
13
5
6
11
10
12
End of generator
```