Josephus problem

This is an old problem from ancient history, but many aspects of it are still unsolved. There are n numbers $1, 2, \ldots, n$ arranged in a circle in clockwise order. Given a number $k \geq 2$, the following step is executed repeatedly, until only one number is left. Move in clockwise direction, jumping over $k-1$ remaining numbers, and eliminate the kth remaining number encountered. Initially start with number 1, so that if $n \geq k$, the first number eliminated is k. The last number that remains is denoted $J(n, k)$ and is called the Josephus number. For example, $J(5, 3) = 4$ as the numbers are eliminated in the order $3, 1, 5, 2$. A program is to be written to compute $J(n, k)$ for given values of n and k.

An easy way of doing this is to just follow the procedure that defines the function. Define a data type called circular_list whose value is a circular list of numbers, with a pointer to a position in the list. An operation ++ is defined to move the pointer one place in the list in clockwise order. You also have an operation that gives the size of the list and an operation to initialize the list with numbers 1 to n. Finally, there is an operation to remove the number that the pointer is pointing to, and a method to return the value.

Then the program for this can be written as

```
int J(int n, int k) {
      circular_list C;
      C.initialize(n);
      while (C.size() > 1) {
           for (int i = 1; i < k; i++) C++;
           C.remove();
      }
      return C.value();
}
```

While this is a simple solution, it is not good enough to evaluate $J(n, k)$ when n and/or k are large. What is $J(1000000000000, 3)$? This method will fail because we cannot even create a list of 1000000000000 numbers. There is a much better way of doing this, when n is very large. Observe that in the first pass over the list, if $n \geq k$, all multiples of k will be eliminated. Instead of removing each one separately, remove all of them at the same time. The problem now reduces to a similar problem with $n - n/k$ numbers, but they are not consecutive numbers. However, given i, we can find the ith number in the new list easily, without going through the whole list. Implement this idea using a recursive function. Note that when n and k are comparable, this may remove only one number from the list. Can you estimate the number of recursive calls made, as a function of n and k? The value of $J(1000000000000, 3)$ is 706915036709.

If you are unable to do this, you can implement the simple method, which should work correctly for small values of n and k, but you will get only half credit for it. The recursive program is actually much simpler and can be written in 10 lines, if done carefully.

Input-Output

Your program should just read in two values n and k using cin, and print out the value of $J(n,k)$ using cout. The limits on n and k are $2 \leq n \leq 10^{18}$, $2 \leq k \leq 10^6$. Note that you will need to use the long long int type (also called just long long) to store n, but int suffices for k. The time required should be a few seconds at most. You may get a segmentation fault with $k = 10^6$ because of the limit on memory for recursive calls. If you are in linux, you can use the command "ulimit -s unlimited" to increase this, or just check with values of $k \leq 10^5$.

Submission

Name your file as RollNo_1.cpp and upload it on moodle. Only the source file should be submitted.

Homework

This method also gives a 'closed form' solution for $J(n,2)$. For example, J(2^n,2) = 1 and J($2^n - 1$,2) = $2^n - 1$ for all $n \geq 1$. Can you find a simple way of computing $J(n,2)$ without using recursion?

A challenging problem is the inverse. Given numbers m and n such that $1 \leq m \leq n$, prove that there always exists a k such that $J(n,k) = m$. Can you write a program to find such a k, given n, m? (This may need some facts that you will learn in CS 207). Prove that for every odd number m, there exists a number n such that $J(n,2) = m$. Is it true that for any number m not divisible by k, there exists an n such that $J(n,k) = m$? More generally, a finite subset of numbers $A \subseteq \{1, 2, \ldots, n\}$ is called a Josephus subset, if for some k, at some stage in the process, the set of remaining numbers is exactly A. It is not true that every subset of $\{1, \ldots, n\}$ is a Josephus subset. For $n = 9$, the subsets $\{1, 2, 5, 8, 9\}$, $\{2, 3, 4, 5, 8\}$, $\{2, 5, 6, 7, 8\}$ cannot be the remaining set of numbers for any choice of k (Why?). There are 13 such sets for $n = 12$, but none for any other values of $n \leq 12$. Can you find other values of n for which such subsets exist? Is it true that every finite subset is a Josephus subset for some n?

To know more about this, and thousands of other such problems, read the book Concrete Mathematics: A Foundation for Computer Science, by Graham, Knuth, Patashnik.