# Recursion

## Explanation of Josephus problem

There are n people standing in a circle waiting to be executed. The counting out begins at some point in the circle and proceeds around the circle in a fixed direction. In each step, a certain number of people are skipped and the next person is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last person remains, who is given freedom. Given the total number of persons n and a number k which indicates that k-1 persons are skipped and kth person is killed in circle. The task is to choose the place in the initial circle so that you are the last one remaining and so survive.

For example, if n = 5 and k = 2, then the safe position is 3. Firstly, the person at position 2 is killed, then person at position 4 is killed, then person at position 1 is killed. Finally, the person at position 5 is killed. So the person at position 3 survives.
If n = 7 and k = 3, then the safe position is 4. The persons at positions 3, 6, 2, 7, 5, 1 are killed in order, and person at position 4 survives.

### Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

The problem has following recursive structure.

```
josephus(n, k) = (josephus(n - 1, k) + k-1) % n + 1
josephus(1, k) = 1
```

**How does this recursion work?** When we kill k-th person, n-1 persons are left, but numbering starts from k+1 and goes in modular way.

(k+1)-th person in the original circle is now first person.
n-th person in the original circle is now (n-k)-th person.
1-st person in the original circle is now (n-k+1)-th person.
(k-1)-th person in the original circle is now (n-1)-th person.

So we add (k-1) to the returned position to handle all cases and keep the modulo under n. Finally we add 1 to the result.

This solution is not easy to think at the first moment.

A simple solution that comes to our mind is

(josephus(n-1, k) + k) % n

We add k because we shifted the positions by k after first killing. The problem with the above solution is, the value of (josephus(n-1, k) + k) can become n and overall solution can become 0. But positions are from 1 to n. To ensure that, we never get n, we subtract 1 and add 1 later. This is how we get

(josephus(n-1, k) + k - 1) % n + 1

Following is simple recursive implementation of the Josephus problem. The implementation simply follows the recursive structure mentioned above.

C++ Java Python3

```c
#include <stdio.h>

int josephus(int n, int k)
{
  if (n == 1)
    return 1;
  else
    /* The position returned by josephus(n - 1, k)
       is adjusted because the recursive call
       josephus(n - 1, k) considers the original
       position k%n + 1 as position 1 */
    return (josephus(n - 1, k) + k-1) % n + 1;
}

// Driver Program to test above function
int main()
{
  int n = 14;
  int k = 2;
  printf("The chosen place is %d",
                 josephus(n, k));
  return 0;
}

```

Time Complexity: O(n)