CS010C

Lab2

Lab2

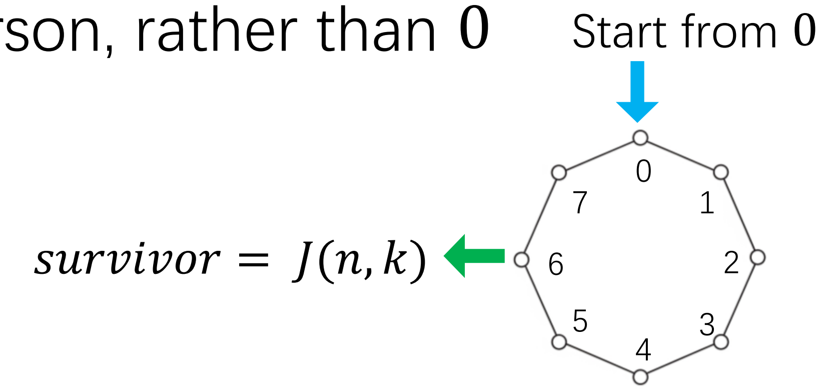
- IntList.h
- main.cpp: No description?
  - I guess it doesn't matter
  - Just return 0 in `int main()` function is OK
- IntList.cpp: I/O
  - Only print blank space between each integer

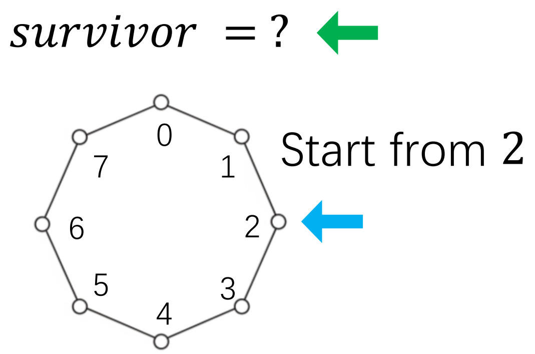
Program1

- Explanation
  - Initial circle n
    - only count first n people in this problem(rather than this “round”)
  - Stride k
    - Begin at next one: count k-th person = skip $k-1$ person
    - Begin at themselves: count $k+1$-th person = skip k person
- Solution1: Doubly-linked list -> circular-linked list
  - More intuitive
    - Resize the circle to n
    - Remove the nodes one-by-one
- Wait, is there any mathematical solution?
  - Think for a while

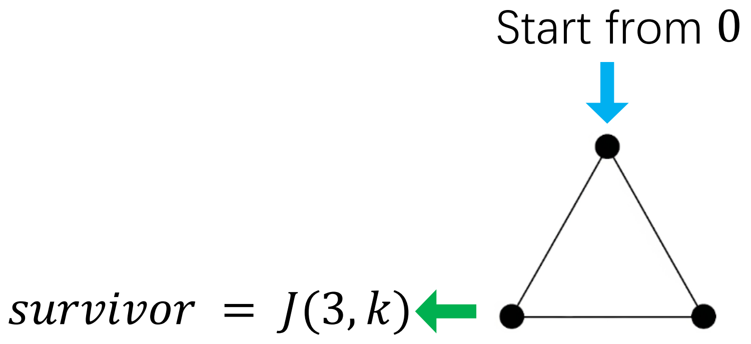
Josephus Problem

- Definition: $J(n,k)$ represents the index of the survivor
  - n is the number of people, k is the number k-th person to vote out
  - $J(n,k) \in [0, n-1]$ (numbering starts from 0, and the last person is $n-1$)
- Puzzle0: Corner cases
  - Case: $k = 0$ => $J(n,0) = n-1$ (the last person survives)
  - Case: $n = 1$ => $J(1,k) = 0$ (game ends)
- Puzzle1: Rotation
  - Start from b-th person, rather than 0
  - It's a circle!
  - $survivor \ = ?$



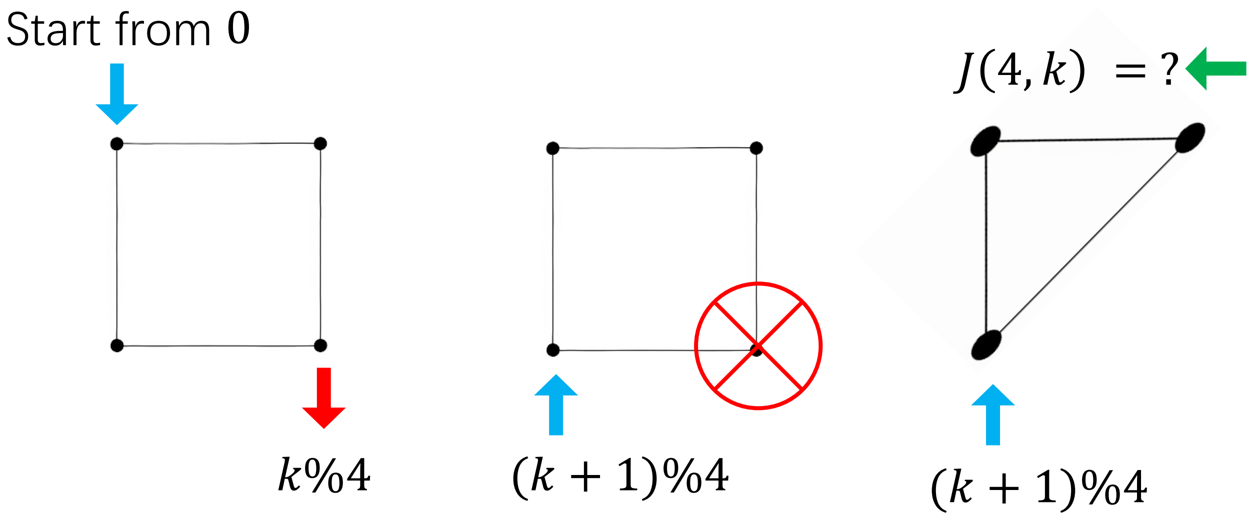


Josephus Problem

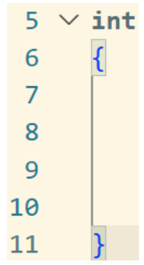


- Puzzle2: Look in reverse
  - Given k ,
    - $J(1,k) = 0 \ \Rightarrow\ J(2,k) = ?$
    - $J(2,k) = 0 \ \Rightarrow\ J(3,k) = ?$

- Try to retrieve the person
  - instead of voting them out
- $J(i,k) = 0 \ \Rightarrow\ J(i+1,k) = ?$

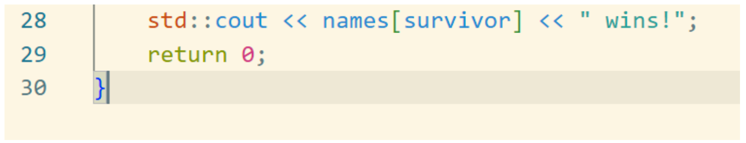


- Solution2: Computing $J(n,k)$
  - More efficient (only arithmetic is involved — no linked list access)
  - The computation process can be simplified to just a few lines!



```
int
{


}
```



```
    std::cout << names[survivor] << " wins!";
    return 0;
}
```