约瑟夫生者死者游戏

#### 2053300 胡锦晖

# 项目概述

## 1.1 项目背景

在一艘有N名旅客的船上，由于船严重超载，必须要抛下一些乘客，使船上只剩下K名乘客(K < N)。制定如下规则进行筛选：所有旅客围成一圈编号1-N，现从第S名旅客开始，按照编号递增的顺序从1开始报数，报到M的人被抛弃，然后从被抛弃的下一位开始重新从1开始报数，重复上述过程直到船上只剩下K个人为止。

## 1.2 项目目标

通过模拟方法找出所有需要抛弃的乘客的编号，并最后输出最后剩余的乘客编号。

## 1.3 项目功能

### （1）输入及错误处理

根据提示分别输入总人数N、起始位置S、死亡数字M以及剩余人数K，其中：

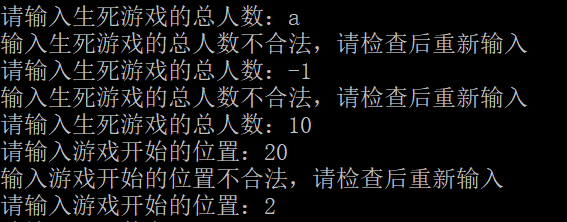
N，M的取值为[1, INT\_MAX]

S，K的取值为[1, N]

错误处理策略：

即将用input函数处理四个待输入数据，当且仅当cin状态正确且data处于由low、high限定的合法范围内时才结束一次输入

错误处理实例：



### （2）输出

输出每一次报数的死者位置，并在最后输出剩余的生者人数及位置信息。

# 实现思路

## 2.1 数据的存储结构

按照题目要求，采用**单循环链表**模拟N名乘客围成一圈。

## 2.2 类的设计及关系

图示

中度可信度描述已自动生成

## 2.3 核心代码

# 文本 描述已自动生成三、性能分析

## 3.1 单循环链表

对于用单循环链表模拟的方法求解约瑟夫环的问题，虽然相较于普通单链表而言，具有向下枚举及删除结点不需要考虑头尾的优势，但由于 直接模拟了整个报数的循环过程，如果所给的死亡数字M远大于总人数N的话，在模拟的过程中会进行多次的整圈循环，所以可以用**求余**的方法计算等价的最低枚举次数。

文本

描述已自动生成改进之后的代码为：

## 3.2 有序集合模拟

对于单循环链表的模拟方式，虽然通过求余运算避免了多次整圈循环的过程，但由于链表结构的限制，想要得到需要删除的节点，就必须从当前节点向下枚举得到，对于N值很大的情况，这一枚举过程仍然需要较大的时间开销。

我们可以使用数组结构，可以直接通过下标得到需要删除的位置，为了将这个有序集合模拟成一个循环链表的结构，我们可以假设将这个有序集合的长度扩大若干倍，然后通过求余运算得到真正的位置：

这样既能把之前m过大循环过多的情况解决，又能找到真正的位置，即将这个环先假设成线性的然后再去通过求余找到真正的位置。

## 3.3 递推公式

若将最终所求解的序号记为一个关于总人数以及死亡数字的函数f(n, m)，n为总人数，m为死亡数字，则例如对于总人数为10人、起始序号为1、死亡数字为3的约瑟夫环，求解f(10,3)的过程为：

起始状态：0 1 2 3 4 5 6 7 8 9

第一次： 0 1 3 4 5 6 7 8 9 🡪 3 4 5 6 7 8 9 0 1

即为：(0+3)%10 (1+3)%10 (2+3)%10 ···(7+3)%10 (8+3)%10

而f(9,3)：0 1 2 3 4 5 6 7 8

则有：f(10,3) = (f(9,3) + 3) % 10

图示, 示意图

描述已自动生成可以得到递推公式如下：

文本

描述已自动生成这样我们就不用模拟操作，可以直接从数值的关系找到递推的关系，代码如下：

屏幕上有字

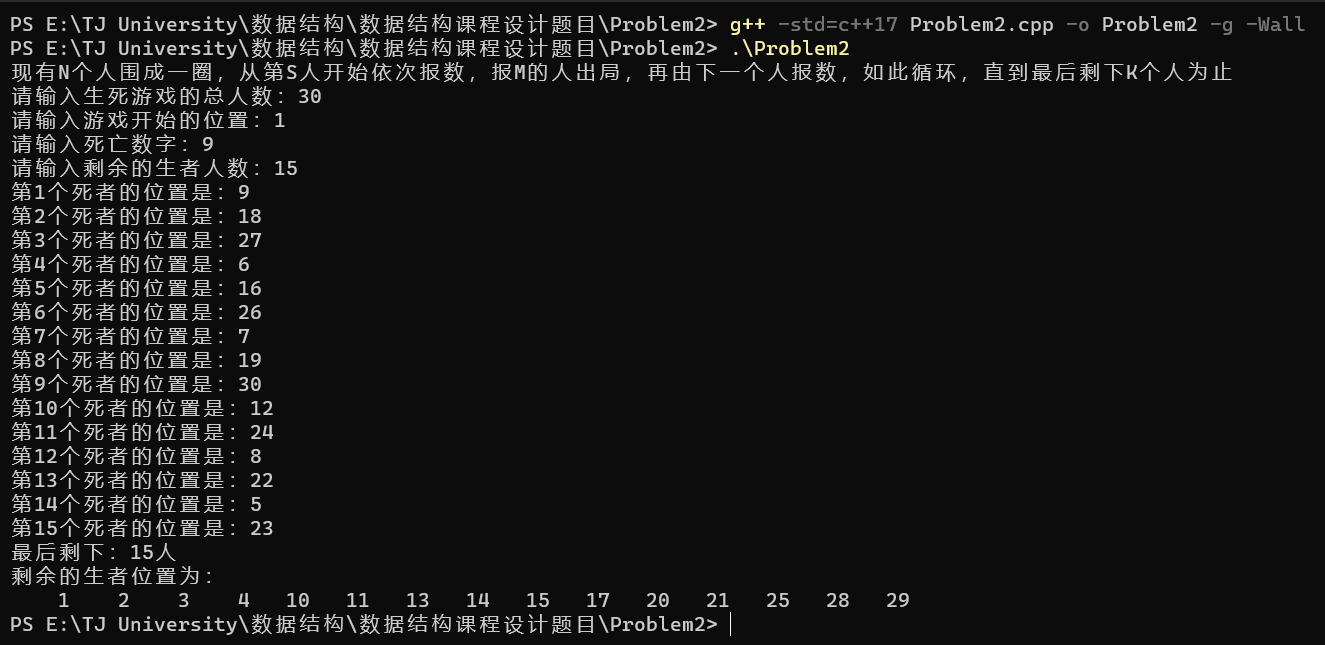
描述已自动生成由于递归效率的问题，也可以写成从前往后的循环：

# 四、测试结果

## 4.1 Windows平台

操作系统：Windows11

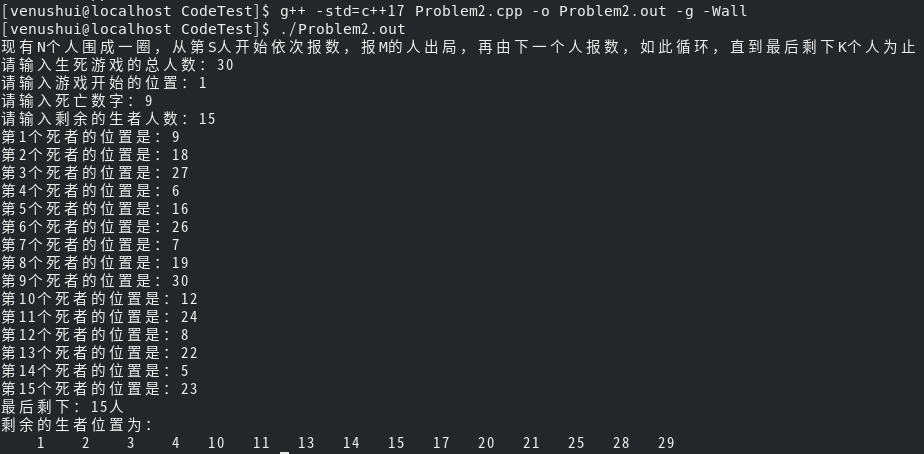
编译器：g++ (GCC) 8.1.0

C++标准：c++17

## 4.2 Linux平台

操作系统：CentOS Linux release 8.5.2111

编译器：g++ (GCC) 8.5.0-4

C++标准：c++17

## 4.3 Mac平台

操作系统：macOS Monterey 12.0.1

编译器：clang++ 13.0.0

电脑萤幕画面

描述已自动生成C++标准：std=c++17