1. 已知有 N 個人在排隊，每個人的名字都不重複，給你一開始所有人的先後順序 (例如：steven、achin、yang、nkhg)，接下來會有很多操作和詢問交錯。

   操作的形式有以下三種可能：

   (1) 叫某一個人到排頭 (yang 到排頭，就會變成 yang、steven、achin、nkhg)

   (2) 叫排頭的人去排尾

   (3) 叫排頭的人離開隊伍

   試問某一時刻目前排頭是誰？

2. 你需要維護一個字典集，一開始給你 N 個字典中的單字。

   接下來會有 Q 次操作。

   操作有三種方式：(1)新增一個單字 (2)刪除一個單字 (3)查看一個單字是否在字典內

   請問該如何維護？

3. 想像你有一個箱子，需要實作以下四種功能：

   (1) 把箱子變成空的

   (2) 加入一顆寫了某個值 x 的球

   (3) 把寫了某個值 x 的球取出一顆

   (4) 從箱子裡均勻公平獨立的隨機取一顆球出來

   對於最後一個操作，你可以假設我們提供了你一個函數 int randint (int lb, int rb)，他會均勻公平獨立的從[lb, rb]這個區間回傳一個整數。（如果需要，也可以假設 randint 函數的參數跟回傳值都是沒有範圍限制的）

   各組可以為了替自己的資料結構辯護而自由選取：x 的值跟箱子最大 size 的範圍都在 (1) int 範圍內，或是(2)沒有特定範圍

   P.S: 各組也可以自己假設同一個數字 x 是否會被重複加入多次（當然，如果一個作法能解決被重複加入多次的情況，那對於不會被重複加入的情況也要能處理。）

4. 對於一個有很多很多作業死線（deadlines）的德田系大學生勳勳，任何時間點都可能會有以下幾種事情之一發生：(1)公布一個新的作業及其死線 (2)現有的一個死線截止 (3)現有的一個作業更改死線

   要如何決定現在該做哪一個作業呢？通常會有以下幾種策略：

   (1) First In First Out (FIFO)：從最早公布的作業開始寫，直到寫完再寫下一個

   (2) Shortest Job First (SJF)：從剩餘花費時間最短的開始寫（假設勳勳可以精準預估花費的時間）

   (3) Earliest Deadline First (EDF)：從死線最近的開始寫

   (4) Priority Scheduling：從最簡單的開始寫

   勳勳發現只有(1)不會中途改去寫別的作業，而(2)、(3)、(4)都有可能會因為新公布的作業而改變目前該寫的作業，比較有彈性，所以勳勳決定視情況採用(2)、(3)、(4)三種策略之一。

   試問勳勳想在任何時間知道：(1)剩餘花費時間最短的作業 (2)距離死線最近的作業 (3)對勳勳來說難度最低的作業（對於以上每一項，各組都可以為了辯論需求假設只有一個或是可能有多個）

   應該怎麼做呢？