چالش بازگشت

در راه بازگشت مسئول قطار از جاناتان و کریستوفر به خاطر زحماتشان در راه رفت تشکر کرد و از آنها خواست تا کلاس های اورا کمی کامل تر کنند. به آنها کمک کنید تا دوتابع calculateRemainTickets و printRemainTicketsOfWagons را پیاده سازی کنند.

*تابع calculateRemainTickets بلیط های باقی مانده را محاسبه میکند و تابع printRemainTicketsOfWagons; بلیط های باقی مانده ی تک تک واگن های قطار را پرینت میکند.

```
class Tickets{
private :
int wagonId;
int bookedTickets;
int totalTickets;
public:
Tickets(int wagonId,int totalTickets , int bookedTickets);
void setBookedTickets(int b);
int getBookedTickets();
void setTotalTickets(int t);
int getTotalTickets();
void setWagonId(int id);
int getWagonId();
};

class Wagon{
private:
Tickets tickets;
Wagon * nextWagon;
public:
void
setTickets(Tickets tickets);
Tickets getTickets();
void setNextWagon(Wagon *nextWagon);
Wagon * getNextWagon();
int calculateRemainTickets();
};

class Train{
private:
```

```
Wagon * headWagon;
int size;
public:
void pushFront(int b , int t , int id);
bool popFront();
void printWagons();
void printRemainTicketsOfWagons();
int getSize();
int getBookedTicketes();
};
```

ورودی

ابتدا نام دستور که یا push است یا pop. در صورتی که دستور push باشد پس از آن در خطوط بعد ابتدا شماره ی واگن خواهد آمد، سپس تعداد بلیط های رزرو شده ی واگن خواهد آمد، در انتها تعداد کل بلیط های ممکن برای واگن خواهد آمد. در آخر عبارت Finish به معنای اتمام ورود اطلاعات خواهد آمد.

خروجی

تعداد بلیط های باقی مانده ی هر واگن به ترتیب از ابتدای لیست تا انتهای لیست باید پرینت شود.در صورت خالی بودن لیست، عبارت Empty پرینت شود.

مثال

ورودی نمونه ۱

```
Push
1
10
20
Push
2
30
31
Push
3
```

```
1
1
Push
4
5
6
Pop
Finish
```

خروجی نمونه ۱

```
0
1
10
```