FILA

Uma fila (= *queue*) é uma estrutura linear sujeita às operações de **inserção** e **remoção**. Essas operações obedecem à política FIFO (= *first in, first out*): o objeto removido é sempre o que foi inserido há mais tempo.

Diz-se que uma das extremidades de uma fila é o *início* ou *cabeça* (= *head*) e a outra extremidade é o *fim* ou *cauda* (= *tail*) da fila. Objetos são inseridos no fim da fila e são removidos do início da fila.

IMPLEMENTAÇÂO USANDO VETOR

```
#define MAXN 1000

typedef struct{

	int v[MAXN];

	int fi,ff;

}TFila;

void nova(TFila * fila){

	fila->fi = 0;

	fila->ff = 0;

}

void enfila(TFila *fila,int x){

	fila->v[fila->ff++] = x;

}

int desenfila(TFila *fila){

	return fila->v[fila->fi++];

}

int esta_vazia(TFila fila){

	return fila.fi == fila.ff;

}

void mostra(TFila fila){

	printf("fi %d ff %d\n",fila.fi,fila.ff);

}

int main(){

  TFila fila;

	nova(&fila);

	enfila(&fila,2);

	enfila(&fila,3);

	while(!esta_vazia(fila)){

		printf("%d\n",desenfila(&fila));

		mostra(fila);

	}
```

```
	return 0;

}
```

IMPLEMENTAÇÃO em C++

```
#include <iostream>

#include <queue>

#include <stdlib.h>

using namespace std;

queue <int> fila;

int main(){

	fila = queue <int> (); //queue não tem o método clear (limpar)

	fila.push(2);//método push coloca um elmento na fila

	fila.push(3);

	fila.push(4);

	cout << fila.size() << endl; //método size retorna o tamanho da fila

	cout << fila.back() << endl; // método back retorna a referencia para o ultimo da fila

	while( !fila.empty() ){//método empty testa se a fila está vazia

		cout << fila.front() << endl;//método front retorna o primeiro elemento da fila

	  fila.pop();//método pop retira o primeiro elemento da fila

	}

	system("PAUSE");

}
```

IMPLEMENTAÇÃO EM JAVA

```
import java.util.Queue;
import java.util.LinkedList;
import java.util.Scanner;

public class FilaDemo {
	public static void main(String args[] ){
		int n;
		Queue<Integer> fila = new LinkedList<Integer>();
		Scanner scan = new Scanner( System.in );
		n = scan.nextInt();
		fila.add(2);
		fila.add(3);
		fila.add(4);
		while( !fila.isEmpty() ){
			System.out.println( fila.remove() );

		}

	}

}
```

10935 - Throwing cards away I

Time limit: 3.000 seconds

Given is an ordered deck of *n* cards numbered 1 to *n* with card 1 at the top and card *n* at the bottom. The following operation is performed as long as there are at least two cards in the deck:

> Throw away the top card and move the card that is now on the top of the deck to the bottom of the deck.

Your task is to find the sequence of discarded cards and the last, remaining card.

Each line of input (except the last) contains a number $n \leq 50$. The last line contains 0 and this line should not be processed. For each number from the input produce two lines of output. The first line presents the sequence of discarded cards, the second line reports the last remaining card. No line will have leading or trailing spaces. See the sample for the expected format.

Sample input

```
7
19
10
6
0
```

Output for sample input

```
Discarded cards: 1, 3, 5, 7, 4, 2
Remaining card: 6
Discarded cards: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 4, 8, 12, 16, 2, 10, 18, 14
Remaining card: 6
Discarded cards: 1, 3, 5, 7, 9, 2, 6, 10, 8
Remaining card: 4
Discarded cards: 1, 3, 5, 2, 6
Remaining card: 4
```

10940 - Throwing cards away II

151 - Power Crisis

Time limit: 3.000 seconds

During the power crisis in New Zealand this winter (caused by a shortage of rain and hence low levels in the hydro dams), a contingency scheme was developed to turn off the power to areas of the country in a systematic, totally fair, manner. The country was divided up into *N* regions (Auckland was region number 1, and Wellington number 13). A number, *m*, would be picked `at random', and the power would first be turned off in region 1 (clearly the fairest starting point) and then in every m'th region after that, wrapping around to 1 after *N*, and ignoring regions already turned off. For example, if $N = 17$ and $m = 5$, power would be turned off to the regions in the order:1,6,11,16,5,12,2,9,17,10,4,15,14,3,8,13,7.

The problem is that it is clearly fairest to turn off Wellington last (after all, that is where the Electricity headquarters are), so for a given *N*, the `random' number *m* needs to be carefully chosen so that region 13 is the last region selected.

Write a program that will read in the number of regions and then determine the smallest number *m* that will ensure that Wellington (region 13) can function while the rest of the country is blacked out.

Input and Output

Input will consist of a series of lines, each line containing the number of regions (*N*) with $13 \leq N < 100$. The file will be terminated by a line consisting of a single 0.

Output will consist of a series of lines, one for each line of the input. Each line will consist of the number *m* according to the above scheme.

Sample input

```
17
0
```

Sample output

```
7
```

440 - Eeny Meeny Moo