**INOI, Iranian National Olympiad in Informatics**
**Online Contest, April 2008**
**Fifth Exam**

# Click

The graph $G$, has $n$ vertices and $m$ edges. This graph has no loops (edges with identical endpoints) and no multiple edges (edges with identical endpoints). In other words $G$ is simple. The vertices of $G$ are numbered from 0 to $(n-1)$. On each edge of $G$ a single integer between 0 and $k-1$ (inclusive) has been written. You have access to this graphs information via a library and you can change integers written on the edges in a certain way that will be explained. In this library three functions named `libN`, `libK` and `click` have been implemented so that you can interact with the graph. The function `click(i)` clicks on the $i^{th}$ vertex. By doing this, the value of all edges connected to vetex $i$ increase by one, except edges that are valued $k-1$ which change to 0 (0 to 1, 1 to 2, ... $k-1$ to 0). The total number of edges whos values have turned into zeros will be returned by this function. The runtime of function `click(i)` is proportional to the number of edges connected to vertex $i$. In other words this function takes $\Theta(d_i)$ time, where $d_i$ is degree of vertex $i$.

The graphs have been embedded into the library and your program shouldn't read anything from the *Standard Input*. Eleven graphs numbered from 0 to 10 have been designed within the library. To load the $t^{th}$ graph you should call `reset(i)`.

You job is to use the library given to you, to determine the set of edges of each of the eleven graphs, and the numbers initially written on them. Notice that the `click` function alters the numbers initially written on the graph's edges.

We expect you to store the required output in eleven files named `click0.out` ... `click10.out` and put them all into a single folder named `click`. Then you shall archive the folder and send an archive named `click.zip` or `click.tgz`. In case you don't put all the eleven files in the archive mentioned, only those that have been properly stored would be graded.

## Problem

Write a program that

- loads tests and reads $n$ and $k$ from the library,
- uses the library to determine the edges of the graph and the numbers written on them,
- stores the specifications of the initial graph in the related file.

## Output Specification

In the first line you should write $m$ (the number of edges). In each of the next $m$ lines you must describe a single edge of the graph as follows: first write the endpoints of the edge and then write the initial number written on it. Seperate these three integers with single spaces. The order of the endpoints and the order of the edges is of no importance, but edges shouldn't be repeated in the output.

## Grading

If you determine the set of edges correctly but the numbers of one or more of the edges are incorrect, you will gain 40% of the related test case.

## Library

`void reset(int t)`
By calling this function the $t^{th}$ test will be loaded. Before calling this function, by default, the test #0 is loaded. If $0 \leq t \leq 10$ isn't true, this function doesn't do anything.

`int libN()`
By calling this function, $n$ (the number of the graph's vertices) will be returned.

`int libK()`
By calling this function, $k$ will be returned.

`int click(int i)`
If $0 \leq i < n$ this function clicks on vertex $i$ and returns the total number of adjacent edges the get valued as 0 after clicking. Otherwise, nothing will happen to the graph and $-1$ will be returned.

You can download `libclick.h`, `libclick.o`, and `sclick.{c, cpp}` from the download section of the contest system.

The functions you will used have been defined in `click.h`.

To compile your program with `g++` you must copy `libclick.h` and `libclick.o` beside your program (in the same directory) and use the following command: (assuming your program is named `click.cpp`)

```
g++ -O2 -static -lm -o click click.cpp libclick.o
```

By performing this command, an executable file named `click` will be created. The `sclick.{c, cpp}` file is a sample program which demonstrates how to use the library. See the 'interaction' section for further information.

## Restrictions

- $4 \leq n \leq 1000$
- $2 \leq k \leq 4000$
- $1 \leq m \leq 2000$
- The size of the file you send to our server shouldn't exceed 100 kilobytes. Otherwise, your file will be rejected by the server.
- As regards to the fact that you should only send the outputs, your program can take as long as you wish to run. Solving all of the tests can take quite a few minutes.

## Interaction

The following table shows a simple interaction with the library.

| Function call | Description |
|---|---|
| `reset(0)` | Loads the graph #0. |
| `int n = libN();` | Sets `n` equal to the return value, which is 4. |
| `int k = libK();` | Sets `k` equal to the return value, which is 3. |
| `int a = click(0);` | Clicks on vertex 0 and sets `a` equal to 1. |
| `int b = click(3);` | Clicks on vertex 3 and sets `b` equal to 0. |
| `int c = click(1);` | Clicks on vertex 1 and sets `c` equal to 2. |
| `int d = click(2);` | Clicks on vertex 2 and sets `d` equal to 0. |
| `int e = click(3);` | Clicks on vertex 3 and sets `e` equal to 1. |
| `int f = click(1);` | Clicks on vertex 1 and sets `f` equal to 0. |

## Example

A correct solution for test #0 (`click0.out`) is written below. (Although any other order of edges and endpoints are acceptable.)

```
4
1 0 1
2 1 2
3 0 2
3 2 0
```