prisoners

The evil Kang the Penguin has captured a number of $2N$ prisoners! To satisfy his amusement, he is forcing them to play a game. Each prisoner has been assigned an integer ID from 0 to $2N-1$, with no repeats, and he has to find the key corresponding to his ID to escape. However, all the keys are hidden in a puzzle room filled with $2N$ boxes, also labelled from 0 to $2N-1$ with no repeats. Each box contains exactly one key. Every prisoner is allowed to open up to N boxes, one at a time, after which if he does not obtain his or her key, he or she will be sentenced to a lifetime of debugging Linux kernel code in the dungeons. Note that keys are **never** removed from their boxes by the prisoners, even if it is the correct key.

Thankfully, Kang has been careless lately. You have managed to sneak into the dungeons to try and help the prisoners. Once you are done planning strategy with them, you will sneak into the puzzle room and take a look in every box, swapping some of the keys. However, due to the risky nature of the plan, you will not have time to go back to the dungeons to tell the prisoners about the key positions. Also, to avoid detection, you want to swap as few keys as possible, while still saving as many prisoners as you possibly can. Can you foil the evil penguin's scheme?

Scoring

Your score will depend on how many prisoners you save, as well as how many swaps you make, according to the following formula:

$$\frac{\textit{(Proportion of prisoners saved)}^2}{\textit{max(1, Number of swaps made)}}$$

For instance, if you save half the prisoners using 2 swaps, you will score 12.5% of the available points.

Implementation details

You should submit two files for this problem. The first will implement the following procedure:

```
void swapper(int N, int boxes[])
```

It will be provided the following function to use:

```
void swapKeys(int a, int b)
```

The second will implement the following procedure:

```
void prisoner(int N, int id)
```

It will be provided the following function to use:

`int openBox(int x)`

The swapper function will run once first, and will be passed an array indicating which boxes contain which keys (i.e. `boxes[i]` will contain the id of the key in box i). The swapper function is allowed to call the `swapKeys()` function up to 100 times. If it is called more than 100 times, you will score 0 points for the given test case.

After the swapper function returns, the prisoner function will be called $2N$ times, once for each id. The prisoner function is allowed to call the `openBox()` function up to N times. Similarly, if it is called more than N times, you will score 0 points for the given test case.

Your procedure: swapper

`void swapper(int N, int boxes[])`

Description

You will need to implement this function.

It should select a number of keys to swap positions, calling `swapKeys()` the appropriate number of times before returning.

Parameters

- `int N` - See problem statement
- `int boxes[]` - An array with length $2N$, indicating that the box labelled i will contain key `boxes[i]`. Note that no two boxes will contain the same key, and that calling `swapKeys()` will not change this array instance (even though the keys are swapped, the array will still reflect the original values, unless you change it manually).

Grader procedure: swapKeys

`void swapKeys(int a, int b)`

Description

Swaps the keys in boxes `a` and `b` (0-indexed). This procedure may not be called more than 100 times. Note that this procedure will not update the `boxes` array passed to `swapper()` to reflect the new key positions.

Parameters

- `int a` and `int b` - The two boxes whose keys are to be swapped

Your procedure: prisoner

`void prisoner(int N, int id)`

Description

You will need to implement this function. It will be called $2N$ times, once for each id from 0 to $2N - 1$.

It should call `openBox()` up to N times, in order to find the key numbered `id`.

Parameters

- `int N` - See problem statement
- `int id` - The id of the key which needs to be found.

Grader procedure: openBox

`int openBox(int x)`

Description

Opens the box numbered `x` (0-indexed), returning the number of the key inside. This procedure may not be called more than N times for each instance of `prisoner()`. You should attempt to open the box with the correct id within these N tries for each prisoner. Note that keys are **never** removed from their boxes, even if found by the correct prisoner.

Parameters

- `int x` - The box to open.

Return value

Returns the number of the key contained in the opened box.

Subtasks

Subtask 1 (27 points)

N will satisfy $1 \leq N \leq 5$.

Subtask 2 (29 points)

N will satisfy $1 \leq N \leq 50$.

Subtask 3 (44 points)

N will satisfy $1 \leq N \leq 1000$.

Sample Interaction

Swapper

`swapper()` is called with `N` $= 1$ and `boxes` $= \{1, 0\}$.

`swapper()` calls `swapKeys(0,1)` and returns. Key 0 is now in box 0 and key 1 is in box 1. Note that the `boxes` array passed to `swapper()` is unchanged.

Prisoner 0

`prisoner()` is called with `N` $= 1$ and `id` $= 0$.

`prisoner()` calls `openBox(0)`, which returns 0. `prisoner()` then returns.

Prisoner 1

`prisoner()` is called with `N` $= 1$ and `id` $= 1$.

`prisoner()` calls `openBox(0)`, which returns 0. `prisoner()` then returns.

Sample Scoring

The above interaction would score $\frac{0.5^2}{1} = 25\%$ of the available points, as only prisoner 0 finds the correct key.