Dumpling Robot

Time Limit	Memory Limit
1 second	128 MB

Statement

The year is 1998 and you are the proud entrepreneur of a new restaurant chain: *dumplingsgivemedumplingskevinpleasegivemedumplingskevinpleaseohmanireallywantsomedumplings.com*. Your all-you-can-eat dumpling restaurants are known for their dumpling robots, which autonomously make and serve dumplings.

Your robots can expertly craft the most delicious dumplings, but you've run into a slight problem. Your robots produce dumplings too quickly and your customers' bowls will overflow if you pile too many of them on.

Because of this, you have fitted your robot with a detector that can determine when the customer begins eating a dumpling, as well as which dumpling it is. However, this information is delayed and will only be made available to the robot K timesteps after the event.

Write code for your robot. Your objectives are to:

- Minimise the amount of time it takes for the customer to finish eating
- Minimise the number of dumplings that drop out of the bowl

This is an interactive problem. There are subtasks in which the customers will exhibit different behaviours. You do not need to implement one solution for all of the subtasks.

Implementation

All of the functions will have declarations in `dumpling.h`, which you will need to include in your C++ submission with `#include "dumpling.h"`.

At the beginning of the simulation, the following function is called:

```
void init(int S);
```

where `S` is the subtask number of the given test case. Note that you cannot serve dumplings during this phase.

At each step of the simulation, the following function will be called:

```
void step(int event);
```

The parameter `event` will contain the ID for the dumpling that was consumed since the last call to `step`, or -1 if no dumpling was consumed. **IDs are assigned sequentially.**

Your task is to implement the two functions `init` and `step`.

Library

To serve dumplings, call the following function:

```
int serve();
```

The function `serve` will return an ID for the dumpling that was just served.

Simulation details

Before the simulation starts, the simulator decides on a number N, the total supply of dumplings from the robot, as well as the bowl capacity B.

At each timestep, when `serve` is called and the bowl has available space for dumplings, the dumpling will be added to the bowl and a new ID will be assigned. The IDs of the dumplings are numbered sequentially $(1, 2, 3, \ldots)$. `serve` can be called any number of times per step.

After the ith call to `step`, there is a chance the customer will consume the dumpling **with the smallest ID** from the bowl. The ID of the dumpling will be passed to `step` on the $(i + K)$th call to `step`.

Independently of this, after the ith call to `step`, the bowl capacity may increase by one, decrease by one or stay the same. If the bowl capacity decreases when it is already full, then the **dumpling with the largest ID** will drop out.

If the customer tries to consume a dumpling in between timesteps and there are no dumplings, the customer will be considered *idle* until the next attempt.

Dumplings that drop out will not be consumed by the customer.

After exactly N calls to `serve` have been made, making additional calls to `serve` will return -1 and be ignored by the simulator, indicating that the robot should stop serving dumplings and return. At this point, any dumplings remaining in the bowl will be consumed by the customer until there are none left. The simulation stops when all N calls have been made and there are no dumplings left in the bowl.

Constraints

For each scenario,

- $1 \leq N \leq 1000$
- $1 \leq K \leq 50$.
- The number of timesteps will not exceed 10^5.
- The capacity of the bowl B can fluctuate between 5 and 20.
- The rate at which the customer eats dumplings may change.
- The simulation does not adapt its behaviour to your queries.

Scoring

Your solution will be scored based on the following function:

$$\text{Score} = \min\left(1, \max\left(0, 0.2x, 8.2x - 7.2\right)\right) \times 100$$

and $x = \frac{nt}{NT}$ where

- N is the total number of dumplings
- n is the number of dumplings consumed by the customer (i.e. not dropped out)
- t is the number of timesteps that the customer spent *not idle*
- T is the total number of timesteps.

That is, your score is determined using x on two linear scales where 0 is 0 points, 0.9 is 18 points and 1.0 is 100 points.

Additionally, you will receive a score of zero if:

- your program exits unexpectedly, or
- your program prints anything to the standard output, or
- the simulation reaches the maximum number of timesteps, or
- your program exceeds the time/memory limits.

Your score for a subtask will be the minimum score over all test cases in that subtask, scaled down to the number of points for that subtask.

Example

The following table shows an example interaction. Here, $N = 3$ and $K = 2$. Suppose the starting bowl size is 3.

Robot	Simulation	Explanation
	calls `init(4)`	Simulator initialises the interaction on subtask 4.
returns from `init`		Simulator finishes setting up.
	calls `step(-1)`	Simulator runs first call to step.
calls `serve()`		Robot serves first dumpling.
	`serve()` returns 1	Dumpling ID 1 returned
returns from `step`		Robot finishes its step.
	(silent)	Customer consumes dumpling 1.
	calls `step(-1)`	Simulator runs second call to step.
calls `serve()`		Robot serves second dumpling.
	`serve()` returns 2	Dumpling ID 2 returned
returns from `step`		Robot finishes its step.
	(silent)	Bowl capacity decreases from 3 to 2.
	calls `step(1)`	Simulator (on step) relays that dumpling 1 has been consumed.
returns from `step`		Robot finishes its step.

Robot	Simulation	Explanation
	(silent)	Customer consumes dumpling 2.
	calls `step(-1)`	Simulator runs fourth step.
returns from `step`		Robot finishes its step.
	(silent)	Customer tries to consume a dumpling, but couldn't find one.
	calls `step(2)`	Simulator runs fifth step.
calls `serve()`		Robot serves third dumpling.
	`serve()` returns 3	Dumpling ID 3 returned
calls `serve()`		Robot tries to serve fourth dumpling.
	`serve()` returns -1	No more dumplings; -1 returned
returns from `step`		Robot finishes its step.
	(silent)	Customer consumes dumpling 2.
	End	Simulation ends. Score given is 0.15 since $\frac{nt}{NT} = 0.75$.

Subtasks

Number	Points	Additional constraints
1	30	$K = 1$ and the capacity of the bowl will not change.
2	30	The capacity of the bowl will not change.
3	40	No further constraints.

Sample Grader

The sample package contains a sample grader, which will run a simulation for M timesteps. The input to the grader should be of the form:

- First line containing N, K, M, S and B
- M lines, one for each timestep. Each line should contain two integers a and b, where
  - $a = 0$ for if the customer should not try and consume at that timestep
  - $a = 1$ if the customer should try and consume at that timestep
  - $b = -1$ for if the bowl size should decrease by 1 at that timestep
  - $b = 0$ for if the bowl size should not change at that timestep
  - $b = 1$ for if the bowl size should increase by 1 at that timestep

The grader will also terminate early if no further simulation is required. The sample input for the interaction looks like

```
3 2 5 4 3
1 0
0 -1
1 0
1 0
1 0
```

The package also contains `dumpling.h` and a stub implementation. Compile your grader with:

```
g++ -Wall -static -O2 -o dumpling sample_grader.cpp dumpling.cpp
```