Arrayser

You and your friend are playing your favourite game, called *Arrayser*. Like all great games, Arrayser involves an array A with N elements, initially set as $A[i] = 1$ for all $0 \leq i \leq N-1$. You and your friend alternate turns, with you going first. On each turn:

1. The player selects an index i with $A[i] = 1$.
2. $A[i]$ is set to 0. Additionally,
   - If $i \neq 0$, then $A[i-1]$ is set to 0.
   - If $i \neq N-1$, then $A[i+1]$ is set to 0.

   Note that $A[i-1]$ and $A[i+1]$ may have already been set to 0 in a previous turn.

The game ends when $A[i] = 0$ for all i.

After playing several games, you realise that your friend is playing completely randomly! Specifically, they always select an index i uniformly at random from all indices where $A[i] = 1$.

Unlike your friend, you absolutely love playing Arrayser and want each game to last as long as possible. In this problem, you will play $T = 5000$ games and your objective is to maximise the mean[1] number of turns that each game lasts for.

Implementation Details

In this task, **do not read from standard input nor write to standard output.** Do not interact with any files. Do not implement a `main` function. Instead, begin your program by including the header file `arrayser.h` (`#include "arrayser.h"`) and interact with it as described below.

Functions

Your solution will play T games in a single test case. You must implement `play_arrayser`, which is called once for each game:

```
void play_arrayser(int N);
```

- `N` is the size of the array.
- This function must call `do_turn` once for each of your turns.
- This function must return when the game ends.
- This function will be called T times in a single test case. We recommend resetting any global variables each time it is called.

From `play_arrayser`, you can make calls to `do_turn`:

```
int do_turn(int i);
```

- i must be an integer with $0 \leq i \leq N-1$ where $A[i] = 1$.
- After you perform your turn, your friend will perform their turn. This function will return the index i that your friend randomly chose. If your turn was the final turn of the game, then this function will instead return -1.

If you violate any of the conditions listed above, your program will be judged as incorrect and you will receive 0% of the points for the test case.

[1]The *mean* is the sum of the values divided by the number of values. For example, the mean of $[3, 5, 4, 8]$ is $\frac{3+5+4+8}{4} = \frac{20}{4} = 5$.

Subtasks and Constraints

This problem has one subtask with one test case (note the unusually-high time limit). The case has $T = 5000$ games all with $N = 300$. Let S be the mean number of turns that were played in each game (this includes your turns and your friend's turns).

- If $S > 144.2$, your score will be 100.
- If $137 \leq S \leq 144.2$, your score will be on a linear scale from 40 to 100. That is, your score will be $40 + 60 \times \frac{S-137}{144.2-137}$.
- If $S < 137$, your score will be $40 \times \left(\frac{S}{137}\right)^{10}$.

A table with some values of S is shown below.

S	100	110	120	130	136	138	140	142	144
Points	1.72	4.45	10.63	23.67	37.17	48.33	65	81.67	98.33

Experimentation

In order to experiment with your code on your own machine, first downloaded the provided files `arrayser.cpp`, `arrayser.h` and `grader.cpp`.

You should modify `arrayser.cpp`, which contains a basic example implementation.

Compile your solution with:

```
g++ -std=c++20 -O2 -Wall arrayser.cpp grader.cpp -o arrayser
```

This will create an executable `arrayser` which you can run with `./arrayser`. If you have trouble compiling, please send a message in the Communication section of the contest website.

The provided grader reads from standard input in the following format:

- The first and only line of input contains N and T.

Sample Grader Input

The following input will run $T = 10$ games, each with $N = 300$:

```
300 10
```

Random Seed

The sample grader has an integer value `RANDOM_SEED` defined on line 9. If this variable is changed, then the grader will make different random choices. The grader used for judging will use a different random seed to the sample grader.

Sample Grader Debug Output

The sample grader has an integer value `DEBUG_LEVEL` defined on line 10, which is set to `1` by default:

- If `DEBUG_LEVEL = 0`, then the sample grader will print the mean number of turns used after all T games conclude.
- If `DEBUG_LEVEL = 1`, then after each game the grader will print the number of turns used in that game. It will also print the mean after all games conclude.
- If `DEBUG_LEVEL = 2`, then after each turn the grader will print information about that turn. In particular, the grader will print the index that was chosen by your code and the grader, as well as the current array A. It will also print the mean after all games conclude.