H. Compass (Compass)

Problem Statement

Kiwi is good at solving problems about circles. For example, intersection of circles, union of circles, and union of circular sectors all failed to stump him.

Today, **Kiwi** stumbled upon a grid of size 7001×7001, the center of the grid is $(0, 0)$. Each cell in the grid could be colored either black or white. Initially, every cell is white except the cell at $(0, r)$.

Due to the size of the grid, **Kiwi** can only observe it from the outside, so **Kiwi** doesn't know the value of r, however he's sure that $0 < r \le 3000$.

Kiwi wants to draw a circle with center $(0, 0)$ and radius r. You might be wondering how to draw a circle on a grid, so **Kiwi** shared his definition of a circle.

- For $0 \le x \le \sqrt{r^2 - x^2}$, cell $(x, \lfloor\sqrt{r^2 - x^2}\rfloor)$, $(-x, \lfloor\sqrt{r^2 - x^2}\rfloor)$, $(x, -\lfloor\sqrt{r^2 - x^2}\rfloor)$, $(-x, -\lfloor\sqrt{r^2 - x^2}\rfloor)$ is black.
- For $0 \le y \le \sqrt{r^2 - y^2}$, cell $(\lfloor\sqrt{r^2 - y^2}\rfloor, y)$, $(\lfloor\sqrt{r^2 - y^2}\rfloor, -y)$, $(-\lfloor\sqrt{r^2 - y^2}\rfloor, y)$, $(-\lfloor\sqrt{r^2 - y^2}\rfloor, -y)$ is black.
- All other cells are white.

Kiwi also stumbled upon a magical machine to aid him on this project. This machine can do the following operations:

- `set_col(x, y, len, val)`:
  $val \in \{0, 1\}$, For $0 \le k < len$, if $val = 1$, color $(x, y + k)$ black, otherwise color it white.
- `set_row(x, y, len, val)`:
  $val \in \{0, 1\}$, For $0 \le k < len$, if $val = 1$, color $(x + k, y)$ black, otherwise color it white.
- `not_col(x, y, xout, yout, len)`:
  For $0 \le k < len$, if $(x, y + k)$ is white originally, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `not_row(x, y, xout, yout, len)`:
  For $0 \le k < len$, if $(x + k, y)$ is white originally, color $(x_{out} + k, y_{out})$ black, otherwise color it white.
- `and_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \le k < len$, if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **are both black originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `and_row(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \le k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **are both black originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.
- `or_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \le k < len$, if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **has at least one black cell originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `or_row(x1, y1, x2, y2, xout, yout, len)`:

For $0 \le k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **has at least one black cell originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.

- `xor_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \le k < len$,if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **has exactly one black cell originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `xor_row(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \le k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **has exactly one black cell originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.

The machine reads the color of required cell for each operation first, then modify all the cell at once, that is, if the output overlaps with the input, the original state of the input will be read.

Please note that if the operation **Kiwi** performs causes the machine to read or write to cells outside of the grid, the machine explodes and so does **Kiwi**. If more than 6×10^6 operations are done, the machine also explodes. Although this machine is full of danger, **Kiwi** really wants to draw circles. Having said that, **Kiwi** is tired, so he asked you to tell him how to operate the machine so that regardless of the value of r, the machine draws the perfect circle that he has in mind.

Implementation Details

You must add `#include "Compass.h"` in the first line of your code, and implement the following function.

```
void draw_circle();
```

Your program can call the following functions.

```
void set_col(int x, int y, int len, int val);
void set_row(int x, int y, int len, int val);
void not_col(int x, int y, int xout, int yout, int len);
void not_row(int x, int y, int xout, int yout, int len);
void and_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void and_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void or_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void or_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void xor_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void xor_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);

void debug(int x, int y);
```

- `debug(x, y)` askes the sample grader to output the color of cell (x, y), where 1 is black and 0 is white. in the actual grading process, this function does nothing and does not count as an operation.
- the effect of other functions is described in the statement.
- `val` must be `0` or `1`.
- `x,y,x1,y1,x2,y2,xout,yout` must be integers between $[-3500, 3500]$.
- `len` must be a positive integer.

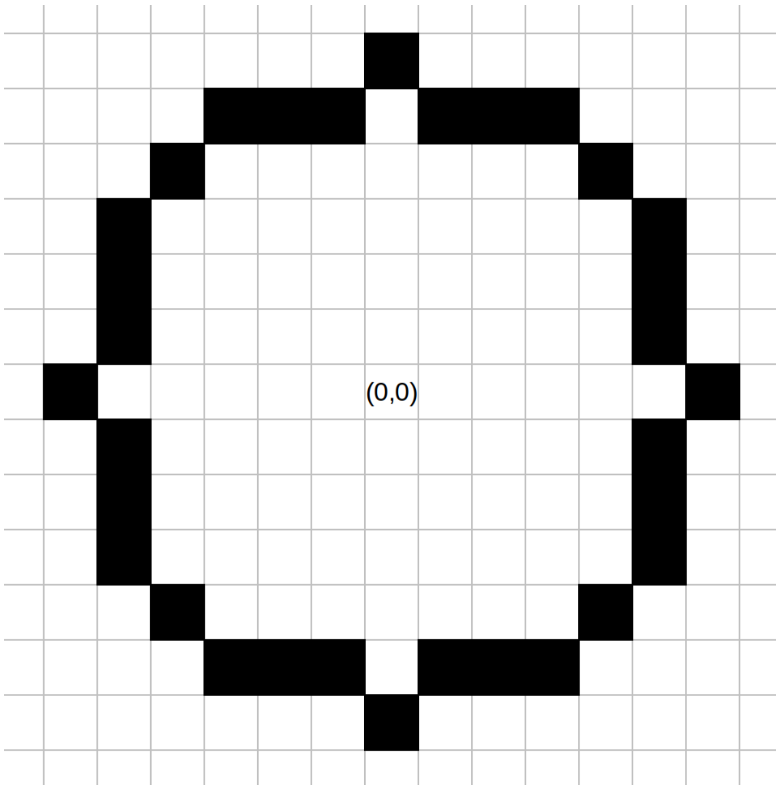
- For `not_col,set_col,and_col,or_col,xor_col`, adding `len-1` to `y,y1,y2,yout` must still be within $[-3500, 3500]$.
- For `not_row,set_row,and_row,or_row,xor_row`, adding `len-1` to `x,x1,x2,xout` must still be within $[-3500, 3500]$.
- You can make at most 6×10^6 calls to functions other than `debug()`.
- The sum of `len` for every call must not exceed 10^9.

Sample Interaction

When $r = 6$, here's a sample program that draws the correct circle:

```
set_col(5, 1, 3, 1);
set_col(5, -3, 3, 1);
or_row(6, 0, 5, 0, 4, 4, 1);
set_row(-3, 5, 7, 1);
and_col(0, 4, 0, 4, 0, 5, 2);
not_row(-4, 4, -4, 4, 1);
xor_col(-5, -3, 5, -3, -5, -3, 7);
xor_col(-5, 0, 6, 0, -6, 0, 1);
set_row(-3, -5, 7, 1);
not_col(0, -6, 0, -6, 2);
set_col(-4, -4, 1, 1);
set_col(4, -4, 1, 1);
```

The end result looks like:



Scoring

There are 3 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	5	$r \le 1000$
2	10	$r \le 2500$
3	85	No other constraints

For the third subtask, if the answer is correct, the machine didn't explode and q operations are done by **Kiwi**, your score for this subtask would be:

- If $q \le 7.5 \times 10^5$, your score for this subtask is 85.
- If $7.5 \times 10^5 < q \le 3.5 \times 10^6$, your score for this subtask is $85 - \frac{55}{2.75\times10^6}(q - 7.5 \times 10^5)$.
- If $3.5 \times 10^6 < q \le 6 \times 10^6$, your score for this subtask is $30 - \frac{30}{2.5\times10^6}(q - 3.5 \times 10^6)$.

Sample Grader

Sample grader reads input with the following format:

- First line: r.

r denotes the radius of the circle.

If your program is judged as Accepted, the sample grader outputs Accepted: q, where q denotes the number of calls to functions other than `debug()`. If your program is judged as Wrong Answer, the sample grader outputs Wrong Answer: MSG, the meaning of MSG is as follows:

- invalid operation: the parameter to function call is invalid, such as going out of bounds when reading or writing.
- too many operations: too many function calls, or the sum of len breaks the constraint.
- wrong result: the final grid isn't what **Kiwi** has in mind.

In the attachment, there's a compressed file named "Compass.zip", when decompressed, there are three folders: `cpp`, `c`, and `examples`. The files in these folders are as follows:

- `cpp`: There is a sample `cpp` file named `Compass.cpp`, you can modify this code or write your own, then use `compile_cpp.sh` or `compile_cpp.bat` to compile on your local computer.
- `c`: There is a sample `c` file named `Compass.c`, you can modify this code or write your own, then use `compile_c.sh` or `compile_c.bat` to compile on your local computer.
- `examples`: the input for sample interaction.