2048

2048 is a sliding block puzzle game created in March 2014 by Gabriele Cirulli. The objective of the game is to slide tiles on a grid and combine them to create a tile displaying the number 2048. Your program must complete a simplified version of a game based on 2048. The rules are as follow:

- The game is played on a 4x4 grid with tiles of varying numbers.
- The tiles on the grid can move left, right, up, and down.
- If two tiles of the same number collide while moving, they merge into a single tile that has the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move.
  For example, two tiles with the value of 4 collide while sliding up and form a tile with the value of 8. If the tile above this newly created 8 tile is also an 8, it does not merge in the same turn.
- Every turn, a new tile will randomly appear in an empty spot with a value of either 2 or 4.
- The game is won when a tile with a value of 2048 appears on the board.
- When there are no empty spaces and no adjacent tiles with the same value, the game ends.

You need to create an AI routine which determines the direction to move the tiles each turn. Your program must be able to beat the game before turn 1,000,000. The game engine has been simplified to remove all randomness but creating the AI routine to win the game is still challenging.

The prototype for the AI routine should be as follows:

In C/C++:

```
int move(int board[4][4]);
```

In Java:

```
public class Q016 {
  // board is guaranteed to be a 4x4 array
  public static int move(int[][] board);
}
```

The `board` parameters specify a 4x4 two-dimensional array. The elements are accessed via row and column indices. If the value of an element is zero, it indicates that there is no tile in the cell. Any value that is not zero indicates the value of the tile that is currently in the cell.

For example, consider the following board:

```
int board[4][4] = {
  {  0,  0,  2,  0 },
  {  2,  0,  0,  0 },
  {  4,  0,  2,  0 },
  {  2,  8,  4,  0 }
}
```

The input above represents the following board status:

The AI routine should return one of the following numbers that tell the game what direction to slide the tiles:

1 - Slide left
2 - Slide right
3 - Slide up
4 - Slide down

The AI routine will be called by the given `main()` function multiple times. In each call, the routine should determine the best direction to move the tiles. The goal of the quest is to win the game within 1,000,000 turns.

The source file you submit will be compiled with a driver module that contains the `main()` function. The driver module drives the game so you only need to provide the AI routine in your submission. If the AI routine you provide wins the game, a special "magic string" will be output, similar to the following:

```
You Win! [******MAGIC STRING******]
```

If the "magic string" is the only output, your submission will be PASSED by the judging system. This also means that your AI routine cannot output anything to standard out.

The following code displays the driver, with the "magic string" removed:

C/C++:

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ROUNDS 1000000
#define THE_GOAL   2048
```

```
extern int move(int board[4][4]);

static void die(const char* msg)
{
 printf("ERROR: %s\n", msg);
 exit(1);
}

static int won(int board[4][4])
{
 int row, col;
 for (row = 0; row < 4; ++row) {
  for (col = 0; col < 4; ++col) {
   if (board[row][col] >= THE_GOAL) {
    return 1;
   }
  }
 }
 return 0;
}

static int next(int board[4][4])
{
 static unsigned long round = 0;
 ++round;
 int empty = 0;
 int row, col;
 for (row = 0; row < 4; ++row) {
  for (col = 0; col < 4; ++col) {
   if (board[row][col] == 0) {
    ++empty;
   }
  }
 }
 if (empty > 0) {
  int tile = 2 * (round % 2 + 1);
  int skip = tile % empty;
  for (row = 0; row < 4; ++row) {
   for (col = 0; col < 4; ++col) {
    if (board[row][col] == 0) {
     --skip;
     if (skip <= 0) {
      board[row][col] = tile;
      return 1;
     }
    }
   }
```

```
  }
 }
 return 0;
}

static void slide(int board[4][4], int direction)
{
 int row, col, pos;
 if (0) {
 } else if (direction == 1) { // LEFT
  for (row = 0; row < 4; ++row) {
   int fix = 0;
   for (col = 1; col < 4; ++col) {
    if (board[row][col] != 0) {
     int now = col;
     for (pos = col - 1; pos >= fix; --pos) {
      if (board[row][pos] == 0) {
       board[row][pos] = board[row][now];
       board[row][now] = 0;
       --now;
      }
      else if (board[row][pos] == board[row][now]) {
       board[row][pos] += board[row][now];
       board[row][now] = 0;
       fix = now;
      }
      else {
       fix = now;
       break;
      }
     }
    }
   }
  }
 } else if (direction == 2) { // RIGHT
  for (row = 0; row < 4; ++row) {
   int fix = 3;
   for (col = 2; col >= 0; --col) {
    if (board[row][col] != 0) {
     int now = col;
     for (pos = col + 1; pos <= fix; ++pos) {
      if (board[row][pos] == 0) {
       board[row][pos] = board[row][now];
       board[row][now] = 0;
       ++now;
      }
      else if (board[row][pos] == board[row][now]) {
       board[row][pos] += board[row][now];
```

```
      board[row][now] = 0;
      fix = now;
     }
     else {
      fix = now;
      break;
     }
    }
   }
  }
 }
} else if (direction == 3) { // UP
 for (col = 0; col < 4; ++col) {
  int fix = 0;
  for (row = 1; row < 4; ++row) {
   if (board[row][col] != 0) {
    int now = row;
    for (pos = row - 1; pos >= fix; --pos) {
     if (board[pos][col] == 0) {
      board[pos][col] = board[now][col];
      board[now][col] = 0;
      --now;
     }
     else if (board[pos][col] == board[now][col]) {
      board[pos][col] += board[now][col];
      board[now][col] = 0;
      fix = now;
     }
     else {
      fix = now;
      break;
     }
    }
   }
  }
 }
} else if (direction == 4) { // DOWN
 for (col = 0; col < 4; ++col) {
  int fix = 3;
  for (row = 2; row >= 0; --row) {
   if (board[row][col] != 0) {
    int now = row;
    for (pos = row + 1; pos <= fix; ++pos) {
     if (board[pos][col] == 0) {
      board[pos][col] = board[now][col];
      board[now][col] = 0;
      ++now;
```

```
      }
      else if (board[pos][col] == board[now][col]) {
       board[pos][col] += board[now][col];
       board[now][col] = 0;
       fix = now;
      }
      else {
       fix = now;
       break; // for (pos)
      }
     }
    }
   }
  }
 } else {
  die("Bad direction.");
 }
}

int main()
{
 int board[4][4] = {
  { 0, 0, 2, 0 },
  { 0, 0, 2, 0 },
  { 0, 0, 0, 0 },
  { 0, 0, 0, 0 },
 };

 int round = 1;
 for (; round <= MAX_ROUNDS; ++round) {
  int direction = move(board);
  slide(board, direction);
  if (won(board)) {
   printf("You Win! [MAGIC STRING]\n");
   exit(0);
  }
  if (next(board) == 0) {
   break; // for(round)
  }
 }
 printf("You Loss!\n");
 return 0;
}
```

Java:

```
public class Task {
 private static final int MAX_ROUNDS = 1000000;
 private static final int THE_GOAL   = 2048;

 private static void die(final String msg) {
  System.out.println(String.format("ERROR: %s",
msg));
  System.exit(1);
 }

 private static boolean won(int[][] board) {
  int row, col;
  for (row = 0; row < 4; ++row) {
   for (col = 0; col < 4; ++col) {
    if (board[row][col] >= THE_GOAL) {
     return true;
    }
   }
  }
  return false;
 }

 static int round = 0;
 private static int next(int[][] board) {
  ++round;
  int empty = 0;
  int row, col;
  for (row = 0; row < 4; ++row) {
   for (col = 0; col < 4; ++col) {
    if (board[row][col] == 0) {
     ++empty;
    }
   }
  }
  if (empty > 0) {
   int tile = 2 * (round % 2 + 1);
   int skip = tile % empty;
   for (row = 0; row < 4; ++row) {
    for (col = 0; col < 4; ++col) {
     if (board[row][col] == 0) {
      --skip;
      if (skip <= 0) {
       board[row][col] = tile;
       return 1;
      }
     }
    }
   }
```

```
  }
  return 0;
 }

 private static void slide(int[][] board, int
direction){
  int row, col, pos;
  if (false) {
  } else if (direction == 1) { // LEFT
   for (row = 0; row < 4; ++row) {
    int fix = 0;
    for (col = 1; col < 4; ++col) {
     if (board[row][col] != 0) {
      int now = col;
      for (pos = col - 1; pos >= fix; --pos) {
       if (board[row][pos] == 0) {
        board[row][pos] = board[row][now];
        board[row][now] = 0;
        --now;
       }
       else if (board[row][pos]==board[row][now]) {
        board[row][pos] += board[row][now];
        board[row][now] = 0;
        fix = now;
       }
       else {
        fix = now;
        break;
       }
      }
     }
    }
   }
  } else if (direction == 2) { // RIGHT
   for (row = 0; row < 4; ++row) {
    int fix = 3;
    for (col = 2; col >= 0; --col) {
     if (board[row][col] != 0) {
      int now = col;
      for (pos = col + 1; pos <= fix; ++pos) {
       if (board[row][pos] == 0) {
        board[row][pos] = board[row][now];
        board[row][now] = 0;
        ++now;
       }
       else if (board[row][pos]==board[row][now]) {
        board[row][pos] += board[row][now];
```

```
       board[row][now] = 0;
       fix = now;
      }
      else {
       fix = now;
       break;
      }
     }
    }
   }
  }
 } else if (direction == 3) { // UP
  for (col = 0; col < 4; ++col) {
   int fix = 0;
   for (row = 1; row < 4; ++row) {
    if (board[row][col] != 0) {
     int now = row;
     for (pos = row - 1; pos >= fix; --pos) {
      if (board[pos][col] == 0) {
       board[pos][col] = board[now][col];
       board[now][col] = 0;
       --now;
      }
      else if (board[pos][col]==board[now][col]) {
       board[pos][col] += board[now][col];
       board[now][col] = 0;
       fix = now;
      }
      else {
       fix = now;
       break;
      }
     }
    }
   }
  }
 } else if (direction == 4) { // DOWN
  for (col = 0; col < 4; ++col) {
   int fix = 3;
   for (row = 2; row >= 0; --row) {
    if (board[row][col] != 0) {
     int now = row;
     for (pos = row + 1; pos <= fix; ++pos) {
      if (board[pos][col] == 0) {
       board[pos][col] = board[now][col];
       board[now][col] = 0;
       ++now;
```

```
      }
      else if (board[pos][col]==board[now][col]) {
       board[pos][col] += board[now][col];
       board[now][col] = 0;
       fix = now;
      }
      else {
       fix = now;
       break;
      }
     }
    }
   }
  }
 } else {
  die("Bad direction.");
 }
}

public static void main(String[] args) {
 int board[][] = new int[][] {
  { 0, 0, 2, 0 },
  { 0, 0, 2, 0 },
  { 0, 0, 0, 0 },
  { 0, 0, 0, 0 },
 };

 int round = 1;
 for (; round <= MAX_ROUNDS; ++round) {
  int direction = Q016.move(board);
  slide(board, direction);
  if (won(board)) {
   System.out.println("You Win! [MAGIC STRING]");
   System.exit(0);
  }
  if (next(board) == 0) {
   break; // for(round)
  }
 }
 System.out.println("You Loss!");
}
}
```