import java.util.Scanner;

public class Ring {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of processes: ");

int num = in.nextInt();

Rr[] proc = new Rr[num];

// Initialize processes

// This code block is initializing the processes. It creates an array of Rr objects with a size of

// `num` (which is the number of processes entered by the user), and then prompts the user to enter

// the ID of each process. It sets the index of each process to its corresponding index in the

// array, sets the state of each process to "active", and sets the value of `f` (which is used as a

// flag during the election process) to 0 for each process.

for (int i = 0; i < num; i++) {

proc[i] = new Rr();

proc[i].index = i;

System.out.println("Enter the ID of process " + (i + 1) + ": ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

// Sort processes based on ID

// This code block is sorting the `proc` array of `Rr` objects based on the `id` field of each

// object. It uses a bubble sort algorithm, where it compares adjacent elements in the array and

// swaps them if they are in the wrong order. The outer loop iterates `num - 1` times, and the

// inner loop iterates `num - 1` times as well. The `if` statement inside the inner loop checks

// if the `id` of the current element is greater than the `id` of the next element. If it is,

// then it swaps the two elements using a temporary variable `temp`. This process continues until

// the array is sorted in ascending order based on the `id` field.

for (int i = 0; i < num - 1; i++) {

for (int j = 0; j < num - 1; j++) {

if (proc[j].id > proc[j + 1].id) {

Rr temp = proc[j];

proc[j] = proc[j + 1];

proc[j + 1] = temp;

}

}

}

// Print the sorted processes

// This code block is printing out the sorted processes in the `proc` array of `Rr` objects. It

// uses a `for` loop to iterate through each element in the array, and prints out the index of

// the element (`i`), the `id` field of the `Rr` object at that index (`proc[i].id`), and a

// space character. The output is formatted as `[index] id `, where `index` is the index of the

// process in the array, and `id` is the ID/name of the process.

for (int i = 0; i < num; i++) {

System.out.print("[" + i + "] " + proc[i].id + " ");

}

// Select last process as coordinator

proc[num - 1].state = "inactive";

System.out.println("\nProcess " + proc[num - 1].id + " selected as coordinator");

// This code block is implementing a loop that repeatedly prompts the user to choose between two

// options: initiating an election or quitting the program. It uses a `while` loop with a

// condition of `true`, which means that the loop will continue indefinitely until it is

// explicitly broken out of using a `return` statement.

while (true) {

System.out.println("\n1. Election\n2. Quit");

int ch = in.nextInt();

// Reset flags

for (int i = 0; i < num; i++) {

proc[i].f = 0;

}

switch (ch) {

case 1:

System.out.println("Enter the process number that initializes the election: ");

int init = in.nextInt();

int temp2 = init;

int temp1 = init + 1;

int i = 0;

while (temp2 != temp1) {

if (temp1 == num) {

temp1 = 0;

}

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

i++;

}

temp1++;

}

System.out.println("Process " + proc[init].id + " sends a message to Process " + proc[temp1].id);

int max = -1;

// Find maximum ID for coordinator selection

for (int j = 0; j <= i; j++) {

if (max < proc[j].id) {

max = proc[j].id;

}

}

// Select coordinator and update states

System.out.println("Process " + max + " selected as coordinator");

for (int k = 0; k < num; k++) {

if (proc[k].id == max) {

proc[k].state = "inactive";

}

}

break;

case 2:

System.out.println("Program terminated.");

in.close();

return;

default:

System.out.println("Invalid response.");

break;

}

}

}

}

class Rr {

public int index; // To store the index of the process

public int id; // To store the ID/name of the process

public int f;

public String state; // Indicates whether the process is active or inactive

}