**DC\_Expt\_6**

**Program :**

import java.io.\*; // Importing necessary package for input handling

// Moin Mohammed Naik

// 211P030

class BullyAlgo {

  int cood, ch, crash; // Variables for coordinator, choice, and crash count

  int prc[]; // Array to represent process states (1 = alive, 0 = crashed)

  // Method to conduct an election when the coordinator crashes

  public void election(int n) throws IOException {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("\n The Coordinator Has Crashed"); // Indicating coordinator failure

    int flag = 1; // Control variable for election loop

    while (flag == 1) {

      crash = 0; // Reset crash count

      // Count the number of crashed processes

      for (int i1 = 0; i1 < n; i1++)

        if (prc[i1] == 0)

          crash++;

      // If all processes are crashed, terminate election

      if (crash == n) {

        System.out.println("\n\*\*\*All processes are crashed \*\*\*");

        break;

      } else {

        System.out.println("\nEnter the Initiator"); // Get initiating process for election

        int init = Integer.parseInt(br.readLine());

        // Check if the initiator is valid (exists and is not crashed)

        if ((init < 1) || (init > n) || (prc[init - 1] == 0)) {

          System.out.println("\nInvalid Initiator");

          continue;

        }

        // Election process begins from the initiator

        for (int i1 = init - 1; i1 < n; i1++)

          System.out.println("Process " + (i1 + 1) + " called for election");

        System.out.println("");

        // Checking process status (Alive or Dead)

        for (int i1 = init - 1; i1 < n; i1++) {

          if (prc[i1] == 0)

            System.out.println("Process " + (i1 + 1) + " is Dead");

          else

            System.out.println("Process " + (i1 + 1) + " is Active");

        }

        // Identify the new coordinator (highest numbered active process)

        for (int i1 = n - 1; i1 >= 0; i1--) {

          if (prc[i1] == 1) {

            cood = (i1 + 1); // Assign highest numbered active process as coordinator

            System.out.println("\n\*\*\* New Coordinator is " + cood + " \*\*\*\*");

            flag = 0; // Election process ends

            break;

          }

        }

      }

    }

  }

  // Method to handle the Bully Algorithm operations

  public void Bully() throws IOException {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Moin MN (211P030)\nEnter the number of processes:");

    int n = Integer.parseInt(br.readLine());

    prc = new int[n]; // Initialize process states (Array to track which process is alive)

    crash = 0;

    // Set all processes to active (1)

    for (int i = 0; i < n; i++)

      prc[i] = 1;

    cood = n; // Initially, the highest numbered process is the coordinator

    do {

      // Menu for user interaction

      System.out.println("\n\t 1. Crash a process");

      System.out.println("\n\t 2. Recover a process");

      System.out.println("\n\t 3. Display New Coordinator");

      System.out.println("\n\t 4. Exit");

      ch = Integer.parseInt(br.readLine()); // Taking user choice

      switch (ch) {

        case 1 -> {

          // Case to crash a process

          System.out.println("Enter a process to crash ");

          int cp = Integer.parseInt(br.readLine());

          // Validate process ID

          if ((cp > n) || (cp < 1)) {

            System.out.println("Invalid Process! Enter a valid process");

          } else if ((prc[cp - 1] == 1) && (cood != cp)) {

            // If the process is active and is not the coordinator

            prc[cp - 1] = 0; // Mark it as crashed

            System.out.println("\nProcess " + cp + " has been crashed");

          } else if ((prc[cp - 1] == 1) && (cood == cp)) {

            // If the crashed process is the coordinator, start election

            prc[cp - 1] = 0;

            election(n);

          } else {

            System.out.println("\nProcess " + cp + " is already crashed");

          }

        }

        case 2 -> {

          // Case to recover a crashed process

          System.out.println("\nCrashed Processes Are: \n");

          // Display crashed processes

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

            if (prc[i] == 0)

              System.out.println(i + 1);

            crash++;

          }

          System.out.println("Enter The Process You Want To Recover");

          int rp = Integer.parseInt(br.readLine());

          if ((rp < 1) || (rp > n)) {

            System.out.println("\nInvalid Process. Enter A Valid ID");

          } else if ((prc[rp - 1] == 0) && (rp > cood)) {

            prc[rp - 1] = 1;

            System.out.println("\nProcess " + rp + " has recovered");

            cood = rp;

            System.out.println("\nProcess " + rp + " is the new coordinator");

          } else if (crash == n) {

            // If all were crashed, the recovered process becomes coordinator

            prc[rp - 1] = 1;

            cood = rp;

            System.out.println("\nProcess " + rp + " is the new coordinator");

            crash--;

          } else if ((prc[rp - 1] == 0) && (rp < cood)) {

            // Recover process without affecting coordinator

            prc[rp - 1] = 1;

            System.out.println("\nProcess " + rp + " has recovered");

          } else {

            System.out.println("\nProcess " + rp + " is not a crashed process");

          }

        }

        case 3 -> // Case to display the current coordinator

          System.out.println("\nCurrent Coordinator is " + cood);

        case 4 -> // Exit the program

          System.exit(0);

        default -> // Handle invalid inputs

          System.out.println("\nInvalid Entry!");

      }

    } while (ch != 4); // Loop until user chooses to exit

  }

  public static void main(String args[]) throws IOException {

    BullyAlgo ob = new BullyAlgo(); // Create an instance of the class

    ob.Bully(); // Call the Bully algorithm

  }

}

**Outpu**t:

