**Experiment : 6 (a,b)**

**Aim** : **Implement Bully algorithm for leader election.**

**Anjali Giri BI20**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Practical :6A**

import java.io.\*;

class BullyAlgo

{

int cood,ch,crash; int

prc[];

public void election(int n) throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); System.out.println("\nThe Coordinator Has Crashed!");

int flag=1;

while(flag==1)

{

crash=0;

for(int i1=0;i1<n;i1++)

if(prc[i1]==0) crash++;

if(crash==n)

{

System.out.println("\n\*\*\* All Processes Are Crashed \*\*\*"); break;

}

else

{

System.out.println("\nEnter The Intiator"); int

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

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

{

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

}

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

System.out.println("Process "+(i1+1)+" Called For Election"); System.out.println("");

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 In");

}

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

if(prc[i1]==1)

{

cood=(i1+1);

System.out.println("\n\*\*\* New Coordinator Is "+(cood)+" \*\*\*"); flag=0;

break;

}

}

}

}

public void Bully() throws IOException

{

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

System.out.println("Enter The Number Of Processes: ");

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

prc=new int[n];

crash=0;

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

prc[i]=1;

cood=n;

do

{

System.out.println("\n\t1. Crash A Process");

System.out.println("\t2. Recover A Process");

System.out.println("\t3. Display New Cordinator");

System.out.println("\t4. Exit");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1: System.out.println("\nEnter A Process To Crash"); int

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

if((cp>n)||(cp<1)){

System.out.println("Invaid Process! Enter A Valid Process");

}

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

{

prc[cp-1]=0;

System.out.println("\nProcess "+cp+ " Has Been Crashed");

}

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

{

prc[cp-1]=0;

election(n);

}

else

System.out.println("\nProcess "+cp+" Is Already Crashed"); break;

case 2: System.out.println("\nCrashed Processes Are: \n"); 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)

{

prc[rp-1]=1;

cood=rp;

System.out.println("\nProcess "+rp+ " Is The New Coordinator"); crash--;

}

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

{

prc[rp-1]=1;

System.out.println("\nProcess "+rp+" Has Recovered");

}

else

System.out.println("\nProcess "+rp+" Is Not A Crashed Process"); break;

case 3: System.out.println("\nCurrent Coordinator Is "+cood); break;

case 4: System.exit(0);

break;

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

}

}

while(ch!=4);

}

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

{

BullyAlgo ob=new BullyAlgo();

ob.Bully();

}

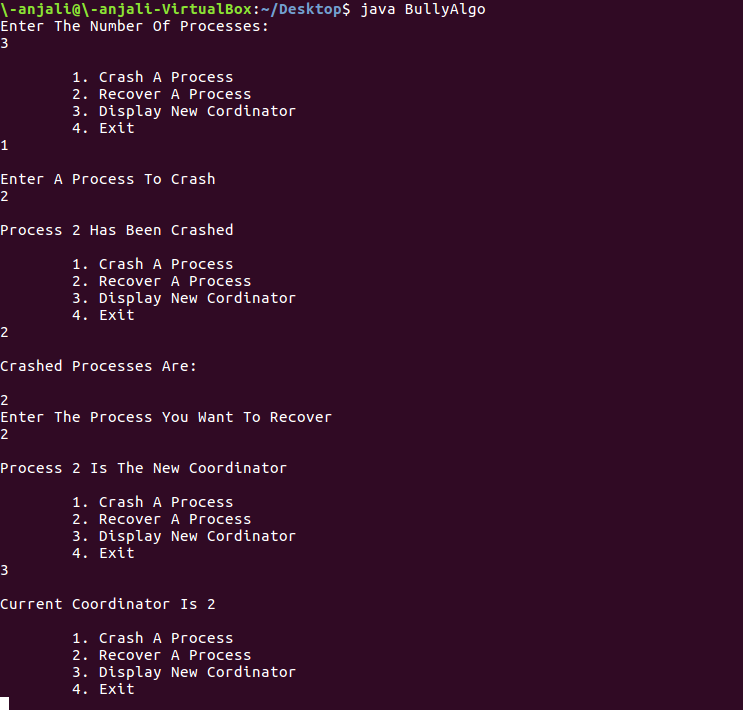
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[c:\javac](file:///C:\c:\javac) BullyAlgo.java

[C:\java](file:///C:\C:\java) BullyAlgo

**Output:**



**Practical 6B:**

**Aim : Implement and Ring algorithm for leader election**.

import java.util.Scanner;

public class Ring {

public static void main(String[] args) {

// TODO Auto-generated method stub

int temp, i, j;

char str[] = new char[10];

Rr proc[] = new Rr[10];

// object initialisation

for (i = 0; i < proc.length; i++)

proc[i] = new Rr();

// scanner used for getting input from console

Scanner in = new Scanner(System.in);

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

int num = in.nextInt();

// getting input from users

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

proc[i].index = i;

System.out.println("Enter the id of process : ");

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

proc[i].state = "active";

proc[i].f = 0;

}

// sorting the processes from on the basis of id

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

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

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

temp = proc[j].id;

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

proc[j + 1].id = temp;

}

}

}

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

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

}

int init;

int ch;

int temp1;

int temp2;

int ch1;

int arr[] = new int[10];

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

System.out.println("\n process" + proc[num - 1].id + "select as co-ordinator");

while (true) {

System.out.println("\n 1.election 2.quit ");

ch = in.nextInt();

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

proc[i].f = 0;

}

switch (ch) {

case 1:

System.out.println("\n Enter the Process number who initialsied election : ");

init = in.nextInt();

temp2 = init;

temp1 = init + 1;

i = 0;

while (temp2 != temp1) {

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

System.out.println("\nProcess " + proc[init].id + "send message to " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

arr[i] = proc[temp1].id;

i++;

}

if (temp1 == num) {

temp1 = 0;

} else {

temp1++;

}

}

System.out.println("\nProcess" + proc[init].id + "sendmessage to " + proc[temp1].id);

arr[i] = proc[temp1].id;

i++;

int max = -1;

// finding maximum for co-ordinator selection

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

if (max < arr[j]) {

max = arr[j];

}

}

// co-ordinator is found then printing on console

System.out.println("\n process " + max + " select as co-ordinator");

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

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

proc[i].state = "inactive";

}

}

break;

case 2:

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

return ;

default:

System.out.println("\n invalid response \n");

break;

}

}

}

}

class Rr {

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

public int id; // to store id/name of process

public int f;

String state; // indiactes whether active or inactive state of node

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Output:**

