**Experiment no: 03**

**Aim: To implement Election Algorithm using Bully Algorithm.**

**Theory:**

In distributed computing, the bully algorithm is a method for dynamically electing a coordinator or leader from a group of distributed computer processes. The process with the highest process ID number from amongst the non-failed processes is selected as the coordinator.

The algorithm uses the following message types:

Election Message: Sent to announce election.

Answer (Alive) Message: Responds to the Election message.

Coordinator (Victory) Message: Sent by winner of the election to announce victory.

When a process P recovers from failure, or the failure detector indicates that the current coordinator has failed, P performs the following actions:

* If P has the highest process id, it sends a Victory message to all other processes and becomes the new Coordinator. Otherwise, P broadcasts an Election message to all other processes with higher process IDs than itself.
* If P receives no Answer after sending an Election message, then it broadcasts a Victory message to all other processes and becomes the Coordinator.
* If P receives an Answer from a process with a higher ID, it sends no further messages for this election and waits for a Victory message. (If there is no Victory message after a period of time, it restarts the process at the beginning.)
* If P receives an Election message from another process with a lower ID it sends an Answer message back and starts the election process at the beginning, by sending an Election message to higher-numbered processes.
* If P receives a Coordinator message, it treats the sender as the coordinator.

**Program:**

#include<stdio.h>

typedef struct process

{

int id;

int crash;

}process;

process P[10];

int total,coordinator;

int highest()

{

int max=0,i,loc;

for(i=1;i<=total;i++)

{

if(P[i].id>max)

{

if(P[i].crash==0)

{

max=P[i].id;

loc=i;

}

}

}

return(loc);

}

void election(int newco)

{

int i,j,new;

int total1=0;

for(j=1;j<=total;j++)

if(P[j].crash!=1)

total1++;

while(newco<=total1)

{

for(i=newco+1;i<=total;i++)

if(P[newco].id< P[i].id)

printf("n Election message sent to Process %d by process %d",i,newco);

printf("n");

for(i=newco+1;i<=total;i++)

if(P[i].crash==0 && P[newco].id< P[i].id )

printf("n OK message received from Process %d",i);

else if(P[i].crash=1 && P[newco].id< P[i].id)

printf("n Process %d is not responding..",i);

newco=newco+1;

if(newco<=total1)

printf("nn process %d is taking over..",newco);

}

coordinator=newco-1;

printf("nn New elected coordinator is Process %d",coordinator);

}

void Crash()

{

int no,i,newco;

printf("n Enter the Process Number of the Process to be crashed: ");

scanf("%d",&no);

P[no].crash=1;

printf("n Process %d has crashed.. ",no);

for(i=1;i<=total;i++)

{

if(P[i].crash==0)

printf("n Process %d is active",i);

else

printf("nn Process %d is Inactive",i);

}

if(no==coordinator)

{

printf("nn Enter a process number which should start the election: ");

scanf("%d",&newco);

election(newco);

}

}

void Recover()

{

int rec;

printf("n Enter the Process number of the process to be recovered: ");

scanf("%d",&rec);

P[rec].crash=0;

election(rec);

}

void Bully()

{

int op;

coordinator=highest();

printf("n Process %d is the Coordinator...",coordinator);

do{

printf("nn 1.Crash n 2.Recover n 3.Exit ");

printf("nn Enter your choice: ");

scanf("%d",&op);

switch(op)

{

case 1: Crash();

break;

case 2: Recover();

break;

case 3:

break;

}

}while(op!=3);

}

int main()

{

int i,id;

int ch;

printf("n Enter Number of Processes: ");

scanf("%d",&total);

printf("nEnter the id for the processes from low priority to high priorityn");

for(i=1;i<=total;i++)

{

printf("Enter id for Process %d: ",i);

scanf("%d",&id);

P[i].id=id;

P[i].crash=0;

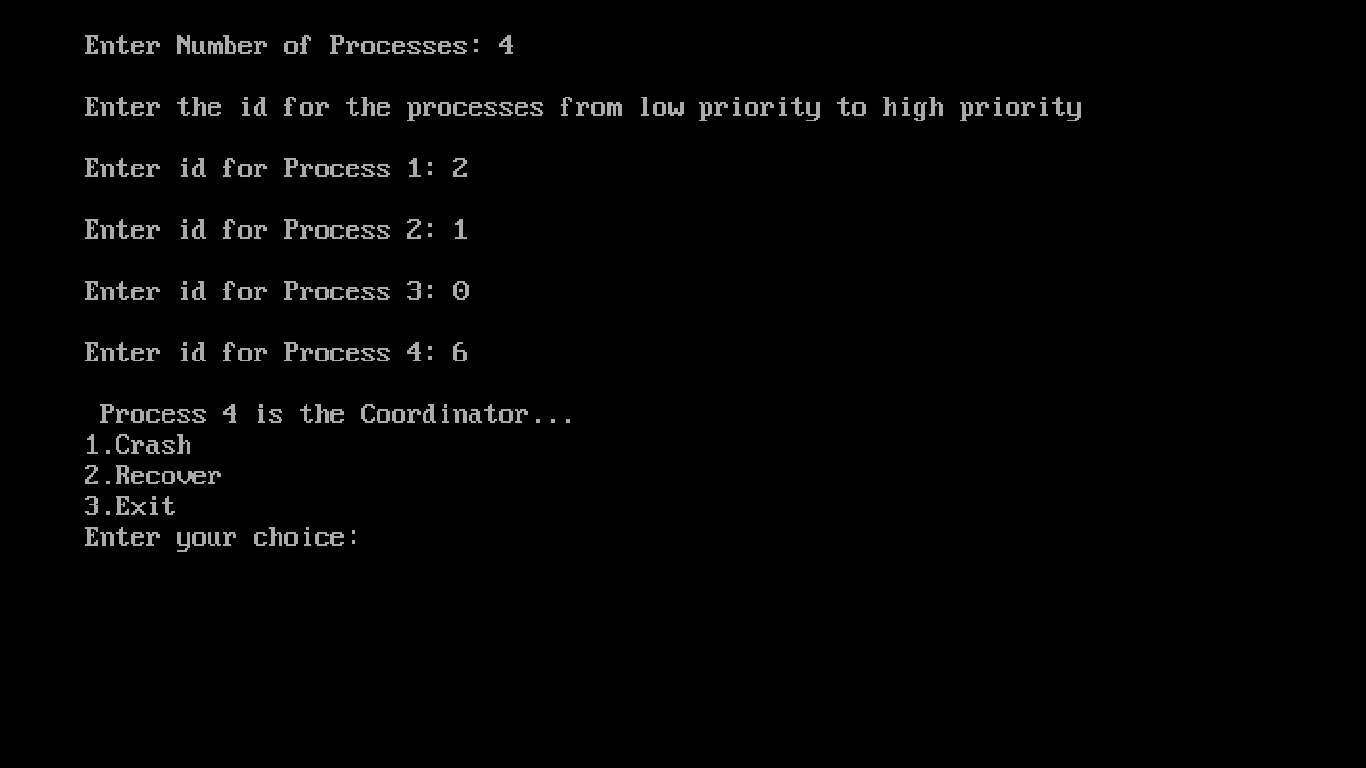
}

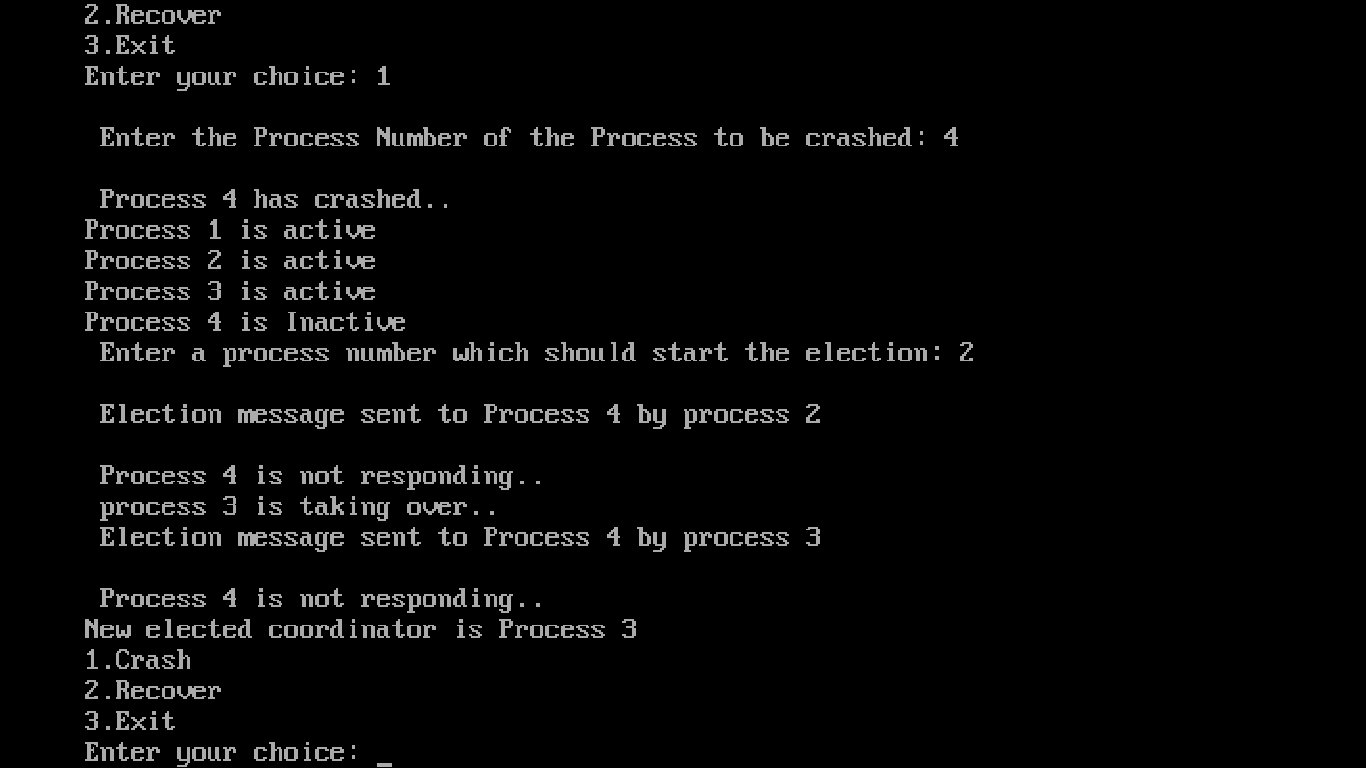
Bully();

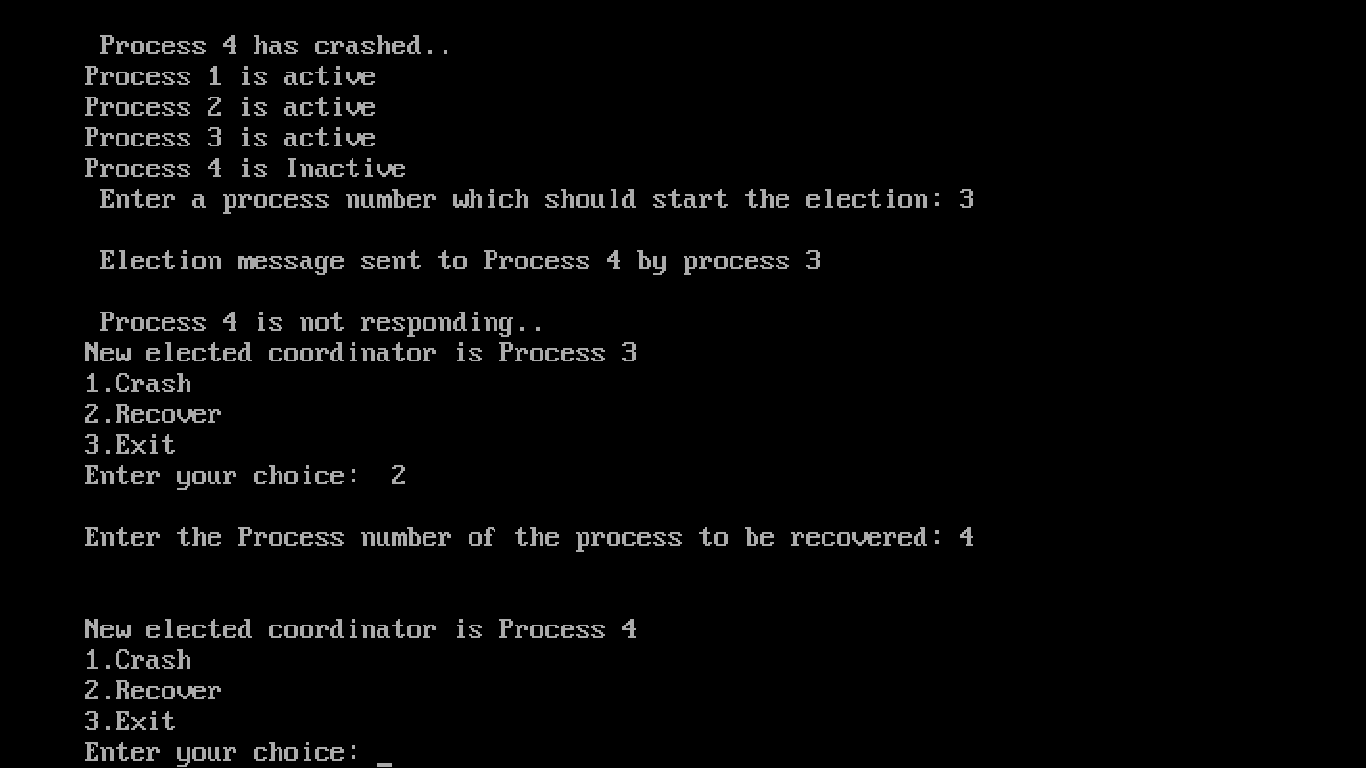
return 0;

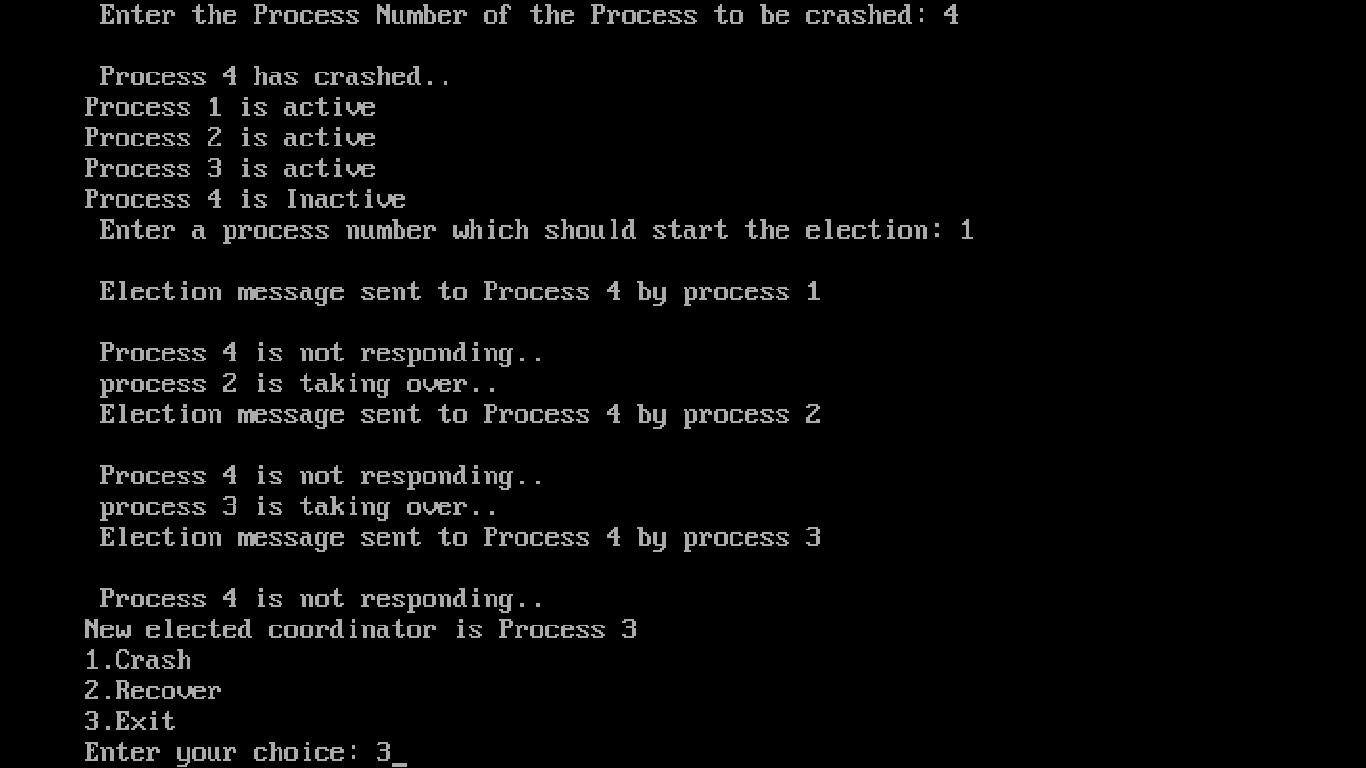
}

**Output:**

****

****



**Conclusion:** Hence we have studied Bully algorithm to implement Election algorithm.