#include<iostream>

using namespace std;

class A{

public:

bool event\_id;

bool signal;

public:

int do\_event\_open(){

cout<<"\nenter event\_id.Event\_id should be 0 or 1\n";

cin>>event\_id;

if(event\_id){

cout<<"event is created. ID is: ";

return event\_id;

}

else{

return -1;

}

}

int do\_event\_close(int eventID,int k){

if(eventID){

cout<<"\n"<<k<<" processes has been created\n.return ";

return k;

}

else{

return -1;

}

}

int do\_event\_wait(int eventID,int k){

for(int i=1;i<=k;i++){

cout<<"\nsignal:\n";

cin>>signal;

if(eventID==signal){ //signal acts as binary semaphore

cout<<"\nevent is signaled.so it will return -1 last\n";

cout<<do\_event\_sig(eventID,i);

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

cout<<"\n\nBlocked p"<<j+1<<" process\n";}

return -1;

}

else{ cout<<"p"<<i<<" process has blocked.return sucess id: 1 \n";

} }

return 1;

}

int do\_event\_sig(int eventID,int k){

cout<<"\np"<<k<<" process running in critical section\n\n";

cout<<"return success id: ";

return 1;

}

};

int main(){

int n;

cout<<"enter the number of processes: ";

cin>>n;

for(int i=1;i<=n;i++){

cout<<"p";

cin>>i;

}

A b;

cout<<b.do\_event\_open();

cout<<b.do\_event\_close(b.event\_id,n);

cout<<b.do\_event\_wait(b.event\_id,n);

return 0;

}