Student Name:	Aditya.Rajesh.Wanwade
Roll No:	C2-27
Practical No:	5
Aim:	Write C programs to simulate solution to Classical Process Synchronization Problem: Dining Philosophers

A) To write C programs to simulate solutions to Dining Philosophers Problem:

```
#include<sys/ipc.h>
#include<stdio.h>
#include<string.h>
#include<sys/msg.h>
#include<stdlib.h>
int one();
int two();
int tph, philname[20], status[20], howhung, hu[20], cho;
int main()
{
int i;
//clrscr();
printf("\n\nDINING PHILOSOPHER PROBLEM");
printf("\nEnter the total no. of philosophers: ");
scanf("%d",&tph);
for(i=0;i<tph;i++)
{
philname[i] = (i+1);
status[i]=1;
}
printf("How many are hungry : ");
scanf("%d", &howhung);
if(howhung==tph)
{
printf("\nAll are hungry..\nDead lock stage will occur");
printf("\nExiting..");
}
else
{
```

```
for(i=0;i<howhung;i++)
{
printf("Enter philosopher %d position: ",(i+1));
scanf("%d", &hu[i]);
status[hu[i]]=2;
}
do
{
printf("1.One can eat at a time\t2.Two can eat at a time\t3.Exit\nEnter your choice:");
scanf("%d", &cho);
switch (cho)
{
case 1: one();
break;
case 2: two();
break;
case 3: exit(0);
default: printf("\nInvalid option..");
}
}
while(1);
}
}
int one()
{
int pos=0, x, i;
printf("\nAllow one philosopher to eat at any time\n");
for(i=0;i<howhung; i++, pos++)
{
printf("\nP %d is granted to eat", philname[hu[pos]]);
for(x=pos;x<howhung;x++)
printf("\nP %d is waiting", philname[hu[x]]);
}
}
int two()
{
```

```
int i, j, s=0, t, r, x;
printf("\n Allow two philosophers to eat at same time\n");
for(i=0;i<howhung;i++)
{
for(j=i+1;j<howhung;j++)
{
if(abs(hu[i]-hu[j])>=1&& abs(hu[i]-hu[j])!=4)
{
printf("\n\ncombination %d \n", (s+1));
t=hu[i];
r=hu[j];
s++;
printf("\nP %d and P %d are granted to eat", philname[hu[i]],
philname[hu[j]]);
for(x=0;x<howhung;x++)
{
if((hu[x]!=t)&&(hu[x]!=r))
printf("\nP %d is waiting", philname[hu[x]]);
}
}
} } }
```

OUTPUT: (All the test cases are included):

```
┌──(kali㉿kali)-[~/C27/lab5]
└─$ vi lab5a.c

┌──(kali㉿kali)-[~/C27/lab5]
└─$ gcc lab5a.c

┌──(kali㉿kali)-[~/C27/lab5]
└─$ ./a.out

DINING PHILOSOPHER PROBLEM
Enter the total no. of philosophers: 5
How many are hungry : 3
Enter philosopher 1 position: 2
Enter philosopher 2 position: 4
Enter philosopher 3 position: 5
1.One can eat at a time 2.Two can eat at a time 3.Exit
Enter your choice:1

Allow one philosopher to eat at any time

P 3 is granted to eat
P 3 is waiting
P 5 is waiting
P 0 is waiting
P 5 is granted to eat
P 5 is waiting
P 0 is waiting
P 0 is granted to eat
P 0 is waiting1.One can eat at a time   2.Two can eat at a time 3.Exit
Enter your choice:2

 Allow two philosophers to eat at same time


combination 1

P 3 and P 5 are granted to eat
P 0 is waiting

combination 2


P 3 and P 0 are granted to eat
P 5 is waiting

combination 3

P 5 and P 0 are granted to eat
P 3 is waiting1.One can eat at a time   2.Two can eat at a time 3.Exit
Enter your choice:3

┌──(kali㉿kali)-[~/C27/lab5]
└─$
```

Result: Thus the program was executed and verified successfully.