Experiment No: 06

Experiment Name: Producer Consumer Problem using Semaphore.

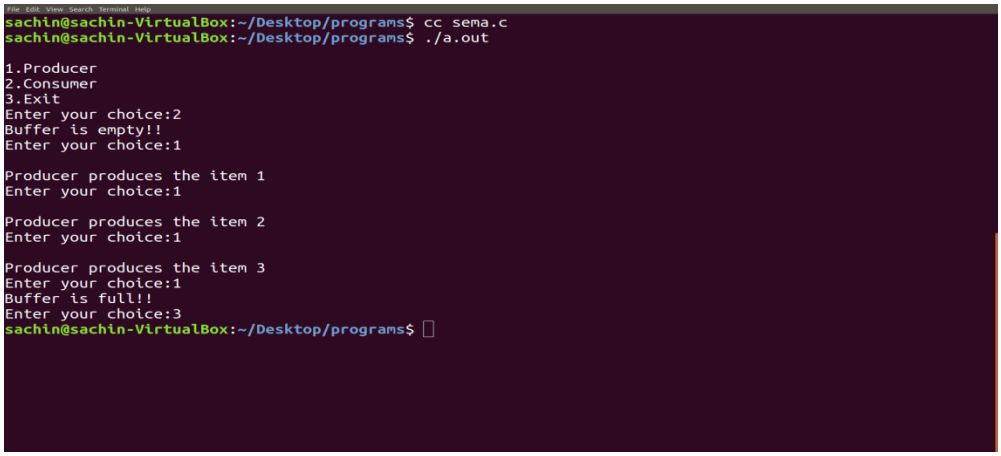
Algorithm:

Source Code:

```
#include<stdio.h>
#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()
{
        int n;
        void producer();
        void consumer();
        int wait(int);
        int signal(int);
        printf("\n1.Producer\n2.Consumer\n3.Exit");
        while(1)
        {
                printf("\nEnter your choice:");
                scanf("%d",&n);
                switch(n)
                {
                        case 1: if((mutex==1)&&(empty!=0))
                                        producer();
                                else
                                        printf("Buffer is full!!");
                                break;
                        case 2: if((mutex==1)&&(full!=0))
                                        consumer();
                                else
                                        printf("Buffer is empty!!");
                                break;
                        case 3:
                                exit(0);
                                break;
                }
        }
```

```
        return 0;
}

int wait(int s)
{
        return (--s);
}

int signal(int s)
{
        return(++s);
}

void producer()
{
        mutex=wait(mutex);
        full=signal(full);
        empty=wait(empty);
        x++;
        printf("\nProducer produces the item %d",x);
        mutex=signal(mutex);
}

void consumer()
{
        mutex=wait(mutex);
        full=wait(full);
        empty=signal(empty);
        printf("\nConsumer consumes item %d",x);
        x--;
        mutex=signal(mutex);
}
```

Output:



VIVA QUESTION:

- What is starvation
- What do you mean by process synchronization.
- What is critical section? What are the requirements to its solution.
- What is semaphore? What are the types of it.

Title: Dead-Lock Avoidance Algorithm

Experiment No: 07

Experiment Name: Dead-Lock Avoidance using Banker's Algorithm.

Algorithm:

1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4….n

2) Find an i such that both
a) Finish[i] = false
b) $Need_i$ <= Work
if no such i exists goto step (4)
3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)

4) if Finish [i] = true for all i
then the system is in a safe state

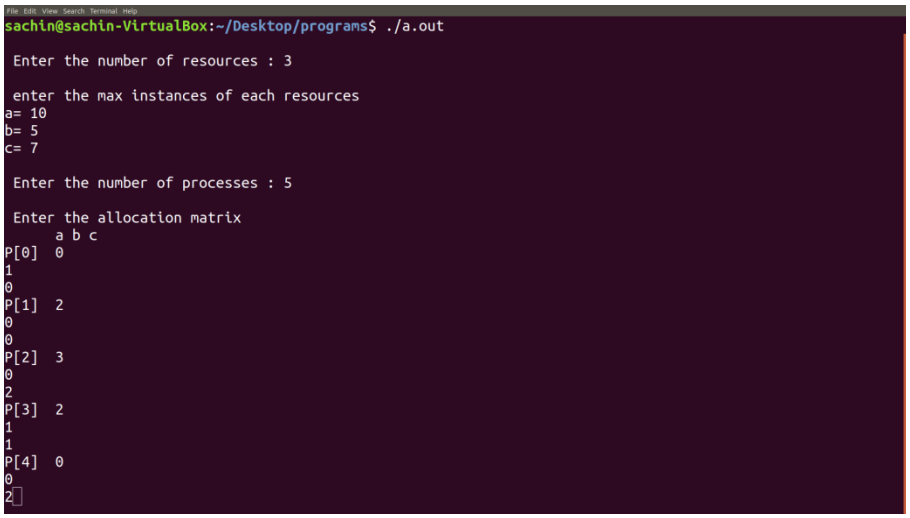
Source Code:

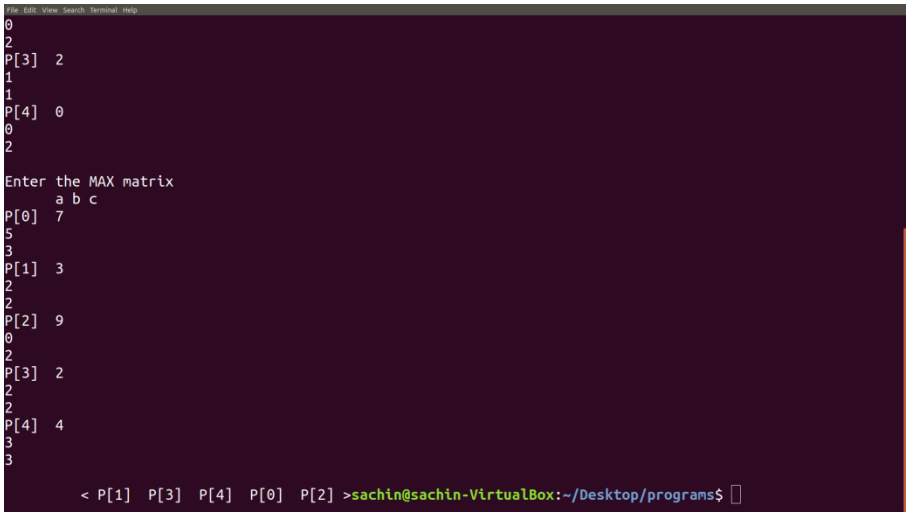
```
#include<stdio.h>
#include<conio.h>
void main() {
```

```
	int
k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,P[10],j
,rz, count=0;

	printf("\n Enter the number of resources : ");
	scanf("%d", &rz);
	printf("\n enter the max instances of each resources\n");
	for (i=0;i<rz;i++) {
		avail[i]=0;
		printf("%c= ",(i+97));
		scanf("%d",&ins[i]);
	}
	printf("\n Enter the number of processes : ");
	scanf("%d", &pno);
	printf("\n Enter the allocation matrix \n     ");
	for (i=0;i<rz;i++)
	printf(" %c",(i+97));
	printf("\n");
	for (i=0;i <pno;i++) {
		P[i]=i;
		printf("P[%d]  ",P[i]);
		for (j=0;j<rz;j++) {
			scanf("%d",&allocated[i][j]);
			avail[j]+=allocated[i][j];
		}
	}
	printf("\nEnter the MAX matrix \n     ");
	for (i=0;i<rz;i++) {
		printf(" %c",(i+97));
		avail[i]=ins[i]-avail[i];
	}
	printf("\n");
	for (i=0;i <pno;i++) {
		printf("P[%d]  ",i);
		for (j=0;j<rz;j++)
		 scanf("%d", &MAX[i][j]);
	}
	printf("\n");
	A: d=-1;
	for (i=0;i <pno;i++) {
		count=0;
		t=P[i];
		for (j=0;j<rz;j++) {
			need[t][j] = MAX[t][j]-allocated[t][j];
			if(need[t][j]<=avail[j])
			 count++;
```

```
                }
                if(count==rz) {
                        output[k++]=P[i];
                        for (j=0;j<rz;j++)
                        avail[j]+=allocated[t][j];
                } else
                 P[++d]=P[i];
        }
        if(d!=-1) {
                pno=d+1;
                goto A;
        }
        printf("\t <");
        for (i=0;i<k;i++)
        printf(" P[%d] ",output[i]);
        printf(">");
        getch();
}
```

Output:





VIVA QUESTION

- Consider two processes P1 and P2. Let each of them have exclusive access to resources R1 and R2. In very simple terms tell me any one scenario that could result in a deadlock?
- What are the necessary conditions for deadlock?
- How the circular wait condition can be prevented .
- What is the drawback of banker's algorithm?
- Assuming the operating system detects the system is deadlocked, what can the operating system do to recover from deadlock?