BANKERS ALGORITHM:

```
kaushik@kaushik-AcerPower-Series:~/OS $ cat banker.c
#include<stdio.h>
#define TRUE 1
#define FALSE 0
void main()
{
int i,j,nProcess,nResource,s,t,flag,Available[10],Max[10][10],Allocation[10][10],Need[10]
[10],Finish[10],Work[10],SafeSeq[10],choice=-1;
printf("\nBANKER'S ALGORITHM\n\n1.Read Data\n2.Print Data\n3.Safety
sequence\n4.Exit");
while(choice!=4)
{
printf("\nEnter an option: ");
scanf("%d",&choice);
switch(choice)
{
case 1://Read Data
printf("\nEnter the number of processes: ");
scanf("%d",&nProcess);
printf("Enter the number of resources available: ");
scanf("%d",&nResource);
printf("\nEnter the number of instances of the resource available:\n");
for(i=0;i<nResource;i++)
{
printf("Resource %c: ",i+65);
scanf("%d",&Available[i]);
}
printf("\nEnter the maximum requirement of each process:\n");
for(i=0;i<nProcess;i++)
{
printf("Process P%d: ",i);
for(j=0;j<nResource;j++)
scanf("%d",&Max[i][j]);
}
printf("\nEnter the allocated instances of resources:\n");
for(i=0;i<nProcess;i++)
{
printf("Process P%d: ",i);
for(j=0;j<nResource;j++)
{
scanf("%d",&Allocation[i][j]);
Need[i][j]=Max[i][j]-Allocation[i][j];
}
}
break;
case 2://Print Data
printf("\nID ALLOCATED MAXIMUM NEED AVAILABLE\n ");
for(i=0;i<4;i++)
{
for(j=0;j<nResource;j++)
```

```
printf("%3c",j+65);
printf(" ");
} for(i=0;i<nProcess;i++)
{
printf("\nP%d ",i);
for(j=0;j<nResource;j++)
printf("%3d",Allocation[i][j]);
printf(" ");
for(j=0;j<nResource;j++)
printf("%3d",Max[i][j]);
printf(" ");
for(j=0;j<nResource;j++)
printf("%3d",Need[i][j]);
if(i==0)
{
printf(" ");
for(j=0;j<nResource;j++)
printf("%3d",Available[j]);
}
}
printf("\n");
break;
case 3://Safety Sequence
for(i=0;i<nProcess;i++)
Finish[i]=FALSE;
for(j=0;j<nResource;j++)
Work[j]=Available[j];
s=0;
flag=TRUE;
while((s<nProcess)&&(flag==TRUE))
{
flag=FALSE;
for(i=0;i<nProcess;i++)
{
if(Finish[i]==FALSE)
{
t=0;
for(j=0;j<nResource;j++)
if(Need[i][j]<=Work[j])
t++;
if(t==nResource)
{
for(j=0;j<nResource;j++)
Work[j]+=Allocation[i][j];
Finish[i]=TRUE;
SafeSeq[s++]=i;
flag=TRUE;
}
}
}
}
if(s==nProcess)
```

```
{
printf("\nThe safety sequence is:\n");
for(i=0;i<nProcess;i++)
printf("P%d ",SafeSeq[i]);
printf("\n");
} else
printf("\nThere is no safety sequence\n");
break;
case 4://Exit
break;
}
}
}
```

```
kaushik@kaushik-AcerPower-Series:~/OS $ gcc -w banker.c
kaushik@kaushik-AcerPower-Series:~/OS $ ./a.out
```

BANKER'S ALGORITHM

1.Read Data
2.Print Data
3.Safety sequence
4.Exit
Enter an option:
1

Enter the number of processes: 5
Enter the number of resources available: 3

Enter the number of instances of the resource available:
Resource A: 3
Resource B: 3
Resource C: 2

Enter the maximum requirement of each process:
Process P0: 7 5 3
Process P1: 3 2 2
Process P2: 9 0 2
Process P3: 2 2 2
Process P4: 4 3 3

Enter the allocated instances of resources:
Process P0: 0 1 0
Process P1: 2 0 0
Process P2: 3 0 2
Process P3: 2 1 1
Process P4: 0 0 2

Enter an option: 2

ID ALLOCATED MAXIMUM NEED AVAILABLE

	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	7	4	3	3	3	2
P1	2	0	0	3	2	2	1	2	2			
P2	3	0	2	9	0	2	6	0	0			
P3	2	1	1	2	2	2	0	1	1			
P4	0	0	2	4	3	3	4	3	1			

Enter an option: 3

The safety sequence is:
P1 P3 P4 P0 P2

Enter an option: 4
kaushik@kaushik-AcerPower-Series:~/OS $