Rupin
20BCE1837
LAB-5

Round Robin with arrival time

```c
#include<stdio.h>

int main()
{

int count,j,n,time,remain,flag=0
int wait_time=0,turnaround_time=
printf("Enter Total Process:\t "
scanf("%d",&n);
remain=n;
for(count=0;count<n;count++)
{
printf("Enter Arrival Time and B
scanf("%d",&at[count]);
scanf("%d",&bt[count]);
rt[count]=bt[count];
}
printf("Enter Time Quantum:\t");
scanf("%d",&time_quantum);
printf("\n\nProcess\t|Turnaround
for(time=0,count=0;remain!=0;)
{
if(rt[count]<=time_quantum && rt
{
time+=rt[count]; rt[count]=0; fl
}
else if(rt[count]>0)
{
rt[count]-=time_quantum; time+=t
}
if(rt[count]==0 && flag==1)
{

remain--;
printf("P[%d]\t|\t%d\t|\t%d\n",c
wait_time+=time-at[count]-bt[cou
}
}
if(count==n-1) count=0;
else if(at[count+1]<=time) co
else
count=0;
}
printf("\nAverage Waiting Tim
printf("Avg Turnaround Time =

return 0;
}
```

```
rupin@rupin:~/Desktop/CSE2005$ gcc roundrobin_with_arrtime.c
rupin@rupin:~/Desktop/CSE2005$ ./a.out
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Process Number 1: 1
5
Enter Arrival Time and Burst Time for Process Process Number 2: 2
6
Enter Arrival Time and Burst Time for Process Process Number 3: 3
7
Enter Arrival Time and Burst Time for Process Process Number 4: 4
8
Enter Time Quantum:     20


Process |Turnaround Time|Waiting Time

P[1]    |       4       |       -1
P[2]    |       9       |       3
P[3]    |       15      |       8
P[4]    |       22      |       14

Average Waiting Time= 6.000000
Avg Turnaround Time = 12.500000rupin@rupin:~/Desktop/CSE2005$
```

Round robin with very long time-quantum-

```
rupin@rupin:~/Desktop/CSE2005$ gcc roundrobin_with_arrtime.c
rupin@rupin:~/Desktop/CSE2005$ ./a.out
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Process Number 1: 1
2
Enter Arrival Time and Burst Time for Process Process Number 2: 2
2
Enter Arrival Time and Burst Time for Process Process Number 3: 3
4
Enter Arrival Time and Burst Time for Process Process Number 4: 4
3
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[1]    |       1       |       -1
P[2]    |       2       |       0
P[3]    |       7       |       3
P[4]    |       7       |       4

Average Waiting Time= 1.500000
Avg Turnaround Time = 4.250000rupin@rupin:~/Desktop/CSE2005$
```

Bankers Algorithm

```c
#include <stdio.h>
#include <stdlib.h>
void final_output(int k[][10], int n, int p)
{
int i, j;
for (i = 0; i < n; i++)
{
printf("\n");
for (j = 0; j < p; j++)
{
printf("%d\t", k[i][j]);
}
}
}
//Banker's Algorithm
void Banker(int A[][10], int N[][10],
int M[10][10], int W[1][10], int *n, int *m)
{
int i, j;
printf("\n Enter total number of processes : ");
scanf("%d", n);
printf("\n Enter total number of resources : ");
scanf("%d", m);
for (i = 0; i < *n; i++)
{
printf("\n Process %d\n", i + 1);
for (j = 0; j < *m; j++)
{
printf(" Allocation for resource %d : ", j + 1);
scanf("%d", &A[i][j]);
printf(" Maximum for resource %d : ", j + 1);
scanf("%d", &M[i][j]);
}
}
printf("\n Available resources : \n");
for (i = 0; i < *m; i++)
{
printf(" Resource %d : ", i + 1);
scanf("%d", &W[0][i]);
}
```

```
41
42 for (i = 0; i < *n; i++)
43 for (j = 0; j < *m; j++)
44 N[i][j] = M[i][j] - A[i][j];
45
46 printf("\n ********Allocation Matrix**********");
47 final_output(A, *n, *m);
48 printf("\n ********Maximum Requirement Matrix" "*********");
49 final_output(M, *n, *m);
50 printf("\n ***********Need Matrix*********");
51 final_output(N, *n, *m);
52 }
53 int safety(int A[][10], int N[][10],int B[1][10], int n, int m, int a[])
54 {
55
56 int i, j, k, x = 0, f1 = 0, f2 = 0;
57 int F[10], W[1][10];
58 for (i = 0; i < n; i++)
59 F[i] = 0;
60 for (i = 0; i < m; i++)
61 W[0][i] = B[0][i];
62
63 for (k = 0; k < n; k++)
64 {
65 for (i = 0; i < n; i++)
66 {
67 if (F[i] == 0)
68 {
69 f2 = 0;
70 for (j = 0; j < m; j++)
71 {
72 if (N[i][j] > W[0][j]) f2 = 1;
73 }
74 if (f2 == 0 && F[i] == 0)
75 {
76 for (j = 0; j < m; j++)
77 W[0][j] += A[i][j]; F[i] = 1;
78 f1++;
79 a[x++] = i;
80 }
81 }
82 }
```

```
}
if (f1 == n)

return 1;
}
return 0;
}

void request(int A[10][10], int N[10][10], int B[10][10], int pid, int K)
{
int rmat[1][10];
int i;
printf("\n Enter additional request : \n");
for (i = 0; i < K; i++)
{
printf(" Request for resource %d : ", i + 1);
scanf("%d", &rmat[0][i]);
}

for (i = 0; i < K; i++)
if (rmat[0][i] > N[pid][i])
{
printf("\n ******Error encountered******\n");
exit(0);
}

for (i = 0; i < K; i++)  if (rmat[0][i] > B[0][i])
{
printf("\n ******Resources unavailable*****\n");
exit(0);
}

for (i = 0; i < K; i++)
{
B[0][i] -= rmat[0][i];
A[pid][i] += rmat[0][i];
N[pid][i] -= rmat[0][i];
}
}
int banker(int A[][10], int N[][10], int W[1][10], int n, int m)
{
int j, i, a[10];
j = safety(A, N, W, n, m, a);
```

```
j = safety(A, N, W, n, m, a);

if (j != 0)
{
printf("\n\n");
printf("\n A safety sequence has been " "detected.\n");
for (i = 0; i < n; i++) printf(" P%d ", a[i]); printf("\n");
return 1;
}
else
{
printf("\n Deadlock has occured.\n"); return 0;
}
}
int main()
{
int All[10][10], Max[10][10], Need[10][10]
, W[1][10];
int n, m, pid, c, r;
printf("\n *********DEADLOCK AVOIDANCE USING" "BANKER'S ALGORITHM**********\n");
Banker(All, Need, Max, W, &n, &m); r = banker(All, Need, W, n, m);
if (r != 0)
{
printf("\n Do you want make an additional" "request for any of the process ? (1=Yes|0=No)");
   scanf("%d", &c);
if (c == 1)
{
printf("\n Enter process number : ");
 scanf("%d", &pid);
request(All, Need, W, pid - 1, m);
r = banker(All, Need, W, n, m);
if (r == 0)
{
exit(0);
}
}
}
else

exit(0);
return 0;
}

```

Output

```
rupin@rupin:~/Desktop/CSE2005$ gcc banker.c
rupin@rupin:~/Desktop/CSE2005$ ./a/out
bash: ./a/out: No such file or directory
rupin@rupin:~/Desktop/CSE2005$ ./a.out

 *********DEADLOCK AVOIDANCE USINGBANKER'S ALGORITHM***********

 Enter total number of processes : 2

 Enter total number of resources : 10

 Process 1
 Allocation for resource 1 : 1
 Maximum for resource 1 : 2
 Allocation for resource 2 : 1
 Maximum for resource 2 : 3
 Allocation for resource 3 : 1
 Maximum for resource 3 : 2
 Allocation for resource 4 : 3
 Maximum for resource 4 : 1
 Allocation for resource 5 : 5
 Maximum for resource 5 : 3
 Allocation for resource 6 : 2
 Maximum for resource 6 : 5
 Allocation for resource 7 : 3
 Maximum for resource 7 : 5
 Allocation for resource 8 : 2
 Maximum for resource 8 : 4
 Allocation for resource 9 : 2
 Maximum for resource 9 : 4
 Allocation for resource 10 : 3
 Maximum for resource 10 : 2

 Process 2
 Allocation for resource 1 : 4
 Maximum for resource 1 : 2
 Allocation for resource 2 : 4
 Maximum for resource 2 : 2
 Allocation for resource 3 : 5
 Maximum for resource 3 : 3
 Allocation for resource 4 : 5
 Maximum for resource 4 : 3
 Allocation for resource 5 : 6
 Maximum for resource 5 : 3
 Allocation for resource 6 : 6
 Maximum for resource 6 : 3
 Allocation for resource 7 : 4
 Maximum for resource 7 : 2
 Allocation for resource 8 : 3
 Maximum for resource 8 : 5
 Allocation for resource 9 : 3
 Maximum for resource 9 : 5
 Allocation for resource 10 : 2
 Maximum for resource 10 : 6

 Available resources :
 Resource 1 : 1
 Resource 2 : 6
 Resource 3 : 2
 Resource 4 : 6
 Resource 5 : 2
 Resource 6 : 7
 Resource 7 : 3
 Resource 8 : 7
 Resource 9 : 4
 Resource 10 : 8

 ********Allocation Matrix**********
1       1       1       3       5       2       3       2       2       3
4       4       5       5       6       6       4       3       3       2
 ********Maximum Requirement Matrix*********
2       3       2       1       3       5       5       4       4       2
2       2       3       3       3       3       2       5       5       6
 ***********Need Matrix*********
1       2       1       -2      -2      3       2       2       2       -1
-2      -2      -2      -2      -3      -3      -2      2       2       4
```

```
 ********Allocation Matrix**********
1       1       1       3       5       2       3       2       2       3
4       4       5       5       6       6       4       3       3       2
 ********Maximum Requirement Matrix*********
2       3       2       1       3       5       5       4       4       2
2       2       3       3       3       3       2       5       5       6
 ***********Need Matrix*********
1       2       1       -2      -2      3       2       2       2       -1
-2      -2      -2      -2      -3      -3      -2      2       2       4


 A safety sequence has been detected.
 P0  P1

 Do you want make an additionalrequest for any of the process ? (1=Yes|0=No)^[[A
```