

Resource Vector: 9 3 6

All the resources can be allocated to process 2

Available resources are: 6 2 3

Process 2 executed: 7

All the resources can be allocated to process 3

Available resources are: 8 3 4

Process 3 executed: 7

All the resources can be allocated to process 4

6/10  
23/8/23

### Dead lock detection

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

```
static int mark[20];
```

```
int i, j, np, nre;
```

```
int main()
```

```
{  
    int alloc[10][10], request[10][10],
```

```
    avail[10], r[10], w[10];
```

```
    printf("Enter the no of process: ");
```

```
    scanf("%d", &np);
```

```
    printf("Enter the no of resources: ");
```

```
    scanf("%d", &nre);
```

```

for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
scanf ("%d", &request[i][j]);
for (i = 0;
for (i = 0; i < n; i++)
{
    printf ("In Total amount of the Resource R is\n");
    scanf ("%d", &n[i]);
    printf ("In Enter the matrix:");
    for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        scanf ("%d", &request[i][j]);
    printf ("In Enter the allocation matrix:");
    for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        scanf ("%d", &alloc[i][j]);
    for (j = 0; j < n; j++)
    {

```



```
avail[i] = alloc[i][j];
```

```
for (i = 0; i < n; i++)
```

```
{  
    int count = 0;
```

```
    for (j = 0; j < n; j++)
```

```
    {  
        if (alloc[i][j] == 0)
```

```
            count++;
```

```
        else
```

```
            break;
```

```
    }  
    if (count == n)
```

```
        mark[i] = 1;
```

```
    }  
    for (j = 0; j < n; j++)
```

```
    {  
        if (request[i][j] <= w[j])
```

```
            can be processed = 1;
```

```
        else
```

```
        {
```

```
            can be processed = 0;
```

```
            break;
```

```
        }  
    }
```

```

if (can be processed)
{
    mark[i] = 1;
    for (j = 0; j < n; j++)
        w[j] += alloc[i][j];
}

```

```

}
}
int deadlock = 0;
for (i = 0; i < n; i++)
    if (mark[i] != 1)
        deadlock = 1;
    if (deadlock)
        printf("No deadlock detected");
    else
        printf("No deadlock possible");
}

```

### Output

Enter the no of processes: 3

Enter the no of resources: 4

Total amount of resource R<sub>1</sub>: 10

Total amount of resource R<sub>2</sub>: 5

Total amount of resource R<sub>3</sub>: 7

Total amount of resource R<sub>4</sub>: 12

line 12

the best time



Enter the request matrix

0	1	0	2
1	0	0	1
1	2	1	0
0	1	0	2
1	0	0	1
1	2	1	0

Enter the allocation matrix:

1	0	0	2
1	1	0	0
0	0	1	1
1	0	0	2
1	1	0	0
0	0	1	1

No deadlock possible

Aim: To write a C program to simulate the following contiguous memory allocation techniques.

a.) Worst fit

Execute Beautify Share Source Code Help

Terminal

```

1  #include<stdio.h>
2  static int mark[20];
3  int i, j, np, nr;
4
5  int main()
6  {
7      int alloc[10][10], request[10][10], avail[10], r[10], w[10];
8
9      printf("Enter the no of processes: ");
10     scanf("%d", &np);
11     printf("Enter the no of resources: ");
12     scanf("%d", &nr);
13     for (i = 0; i < nr; i++)
14     {
15         printf("Total Amount of Resource R %d: ", i + 1);
16         scanf("%d", &r[i]);
17     }
18
19     printf("Enter the request matrix:\n");
20     for (i = 0; i < np; i++)
21     {
22         for (j = 0; j < nr; j++)
23             scanf("%d", &request[i][j]);
24
25         printf("Enter the allocation matrix:\n");
26         for (j = 0; j < nr; j++)
27             scanf("%d", &alloc[i][j]);
28
29         /* Available Resource calculation */

```

```

Enter the no of processes: 3
Enter the no of resources: 4
Total Amount of Resource R 1: 10
Total Amount of Resource R 2: 5
Total Amount of Resource R 3: 7
Total Amount of Resource R 4: 12
Enter the request matrix:
0 1 0 2
1 0 0 1
1 2 1 0
0 1 0 2
1 0 0 1
1 2 1 0
Enter the allocation matrix:
1 0 0 2
1 1 0 0
0 0 1 1
1 0 0 2
1 1 0 0
0 0 1 1
No Deadlock possible

```