Academic Task 3

OPERATING SYSTEM (CSE 316)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

NAME	ROLL No.	REG. No.
ABHINAV AP	K18 JC A20	11803343

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Ques. 20. Consider that a system has P resources of same type. These resources are shared by Q processes time to time. All processes request and release the resources one at a time. Generate a solution to demonstrate that, the system is in safe state when following conditions are satisfied.

Conditions:

1. Maximum resource need of each process is between 1 and P.
2. Summation of all maximum needs is less than P+Q

#CODE

```
#include<stdio.h>

int main()

{

int q, allocation[10], max[10], need[10], available,i,j,flag[10],x=0,sequence[10],c=0;

printf("Enter the number of process \n");

scanf("%d",&q);

for (i=0;i<q;i++)

{

  flag[i]=0;  //initially no process have been allocated resources

}

for(i=0;i<q;i++)

{

  printf("Enter the allocation for process Q[%d]\n",i);

  scanf("%d",&allocation[i]);

  printf("Enter the max for process Q[%d]\n",i);

  scanf("%d",&max[i]);

}

printf("Enter the available resources");

  scanf("%d",&available);

for(i=0;i<q;i++)

{

  need[i]=max[i]-allocation[i];
```

```
}
for(i=0;i<q;i++)
{
   printf("Need of Q[%d] is %d\n",i,need[i]);
}
for(i=0;i<q;i++)
{
   for(j=0;j<q;j++)
   {
      if(flag[j]==0&&need[j]<=available)
      {
      available=available+allocation[j];
      flag[j]=1;
      printf("Process %d has been allocated resources\n",j);
      sequence[c]=j;
      c++;
      }
   }
}
for(i=0;i<q;i++)
{
   if(flag[i]==0)
   {
   printf("system is in unsafe state\n");
   x=1;
   break;
   }
}
if(x==0)
{
```

```
printf("System is in safe state\n");
printf("Safe sequence is \n");
for(i=0;i<q;i++)
{
   printf("Q[%d],",sequence[i]);
}
}
printf("\n");
}
```

PROBLEM IN TERMS OF OPERATING SYSTEM:

Here in this question, it is given that a system has a P resources of same type and it is shared by Q process time to time. And the system should be in safe state when Maximum resource need of each process is between 1 and P and Summation of all maximum needs is less than P+Q.

To find a solution for this problem we must apply Bankers algorithm. The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue

There are certain terms present in this algorithm. Available, maximum demand and need, allocation.

Available indicating the number of available resources of each type. **Maximum resources need** define**s** the maximum demand of each process in a system. **Allocation** defines the number of resources of each type currently allocated to each process. **Maximum need** indicates the remaining resource need of each process.

Here in this question maximum resources ness of each process should be between 1 and P

The summation of maximum need is less than P+Q

Suppose Q process Q1,Q2,Q3…..Qn shares P identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process Qi is Si, where 1< Si< P .

There are Q processes and they share P resources.

Sum of the requirement of all processes will be " Sum of all S_i" i.e. $\sum_{i=1}^{n} Si$. Now If you dont want the deadlock to occur, then you should share P resources such that each process will get 1 less than their maximum requirement and then add 1 resource to any one of the processes. This way you can avoid deadlock.

To ensure deadlock never happens allocate resources to each process in following manner:
Worst Case Allocation (maximum resources in use without any completion) will be (max requirement−1) allocations for each process. i.e., si−1 for each i
Now, if $\sum_{i=1}^{n}(Si - 1) <= P$ dead lock can occur if P resources are split equally among the Q processes and all of them will be requiring one more resource instance for completion.

Now, if we add just one more resource, one of the process can complete, and that will release the resources and this will eventually result in the completion of all the processes and deadlock can be avoided. i.e., to avoid deadlock

$\sum_{i=1}^{n}(Si - 1) + 1 <= P$

$\sum_{i=1}^{n} Si - Q + 1 <= P$

$\sum_{i=1}^{n} Si <= P+Q$

In the given question there is two conditions

1. 1<= max resources <= P
2. $\sum$ Max need <= (P+Q)

ALGORITM

1. Enter the number of process. Which is Q
2. Enter the resource allocated. Which is P

3. Enter the maximum for the process
4. Repeat the process until Qn-1
5. In this case the summation of max need must be less than P+Q
6. Enter the available resource

TIME COMPLEXITY

O(n)+ O(n)+ O(n)+ O(n)

= 4O(n)

=O(n)

BOUNDARY CONDITIONS:

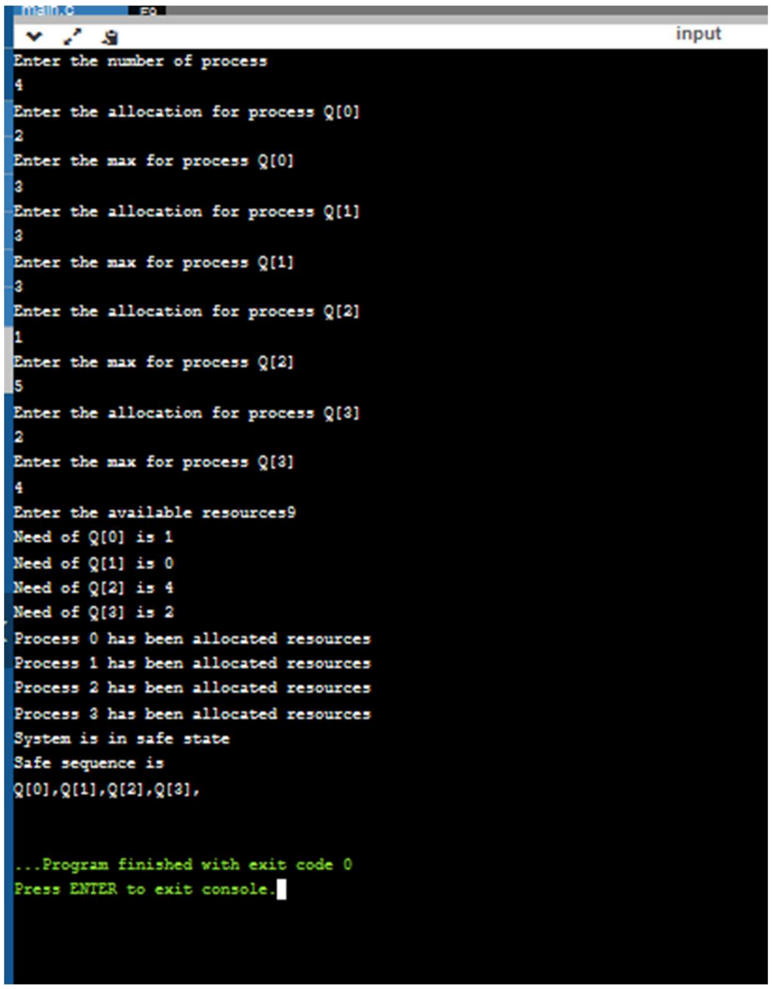
We can give any number of processes as input and we should give resource allocated and the maximum need of each process.

The number of processes should not be negative or zero

TEST CASE

Test case number 1:

Q0	2	3
Q1	3	3
Q2	1	5
Q3	2	4

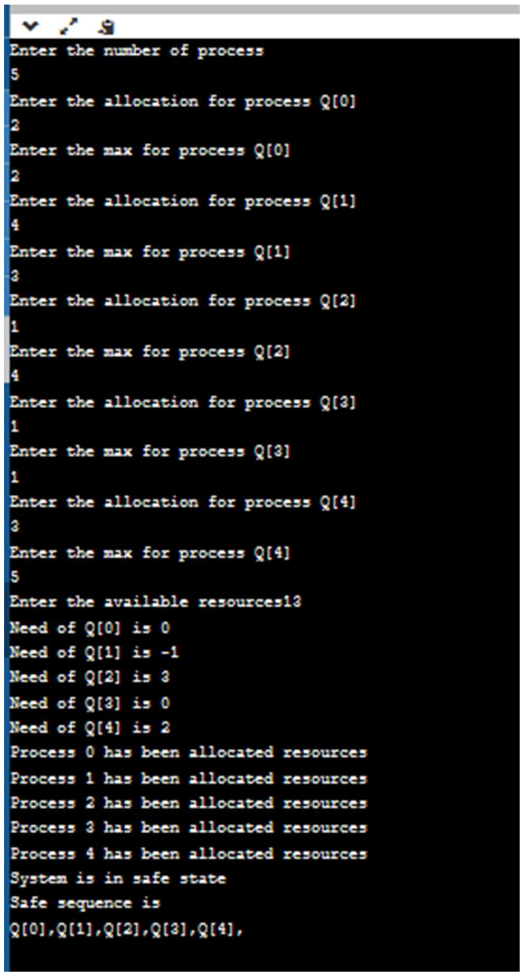


```
Enter the number of process
4
Enter the allocation for process Q[0]
2
Enter the max for process Q[0]
3
Enter the allocation for process Q[1]
3
Enter the max for process Q[1]
3
Enter the allocation for process Q[2]
1
Enter the max for process Q[2]
5
Enter the allocation for process Q[3]
2
Enter the max for process Q[3]
4
Enter the available resources9
Need of Q[0] is 1
Need of Q[1] is 0
Need of Q[2] is 4
Need of Q[3] is 2
Process 0 has been allocated resources
Process 1 has been allocated resources
Process 2 has been allocated resources
Process 3 has been allocated resources
System is in safe state
Safe sequence is
Q[0],Q[1],Q[2],Q[3],


...Program finished with exit code 0
Press ENTER to exit console.
```

Test case number 2:

Q0	2	2
Q1	4	3
Q2	1	4
Q3	1	1
Q4	3	5



```
Enter the number of process
5
Enter the allocation for process Q[0]
2
Enter the max for process Q[0]
2
Enter the allocation for process Q[1]
4
Enter the max for process Q[1]
3
Enter the allocation for process Q[2]
1
Enter the max for process Q[2]
4
Enter the allocation for process Q[3]
1
Enter the max for process Q[3]
1
Enter the allocation for process Q[4]
3
Enter the max for process Q[4]
5
Enter the available resources13
Need of Q[0] is 0
Need of Q[1] is -1
Need of Q[2] is 3
Need of Q[3] is 0
Need of Q[4] is 2
Process 0 has been allocated resources
Process 1 has been allocated resources
Process 2 has been allocated resources
Process 3 has been allocated resources
Process 4 has been allocated resources
System is in safe state
Safe sequence is
Q[0],Q[1],Q[2],Q[3],Q[4],
```

GitHub Link: https://github.com/aby-art/cse316