S.No: 6	Exp. Name: ***Multi-Level Queue scheduling algorithm***	**Date:2025-04-09**

Aim:

Aim: Simulation of Multi-Level Queue scheduling algorithm.

Description:
Multi-level queue scheduling algorithm is used in scenarios where the processes can be classified into groups based on property like process type, CPU time, IO access, memory size, etc. In a multi-level queue scheduling algorithm, there will be 'n' number of queues, where 'n' is the number of groups the processes are classified into. Each queue will be assigned a priority and will have its own scheduling algorithm like round-robin scheduling or FCFS. For the process in a queue to execute, all the queues of priority higher than it should be empty, meaning the process in those high priority queues should have completed its execution. In this scheduling algorithm, once assigned to a queue, the process will not move to any other queues.

Source Code:

MultiLevelQueue.c

```c
#include<stdio.h>
int main()
{
   int p[20],bt[20],su[20],wt[20],tat[20],i,k,n,temp;
   float wtag,tatavg;
   printf("Enter the number of processes: ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
      {
         p[i]=i;
         printf("Enter the Burst Time of Process %d: ",i);
         scanf("%d",&bt[i]);
         printf("System/User Process (0/1): ");
         scanf("%d",&su[i]);
      }
   for(i=0;i<n;i++)
   for(k=i+1;k<n;k++)
   if(su[i]>su[k])
      {
         temp=p[i];
         p[i]=p[k];
         p[k]=temp;
         temp=bt[i];
         bt[i]=bt[k];
         bt[k]=temp;
         temp=su[i];
         su[i]=su[k];
         su[k]=temp;
      }
   wtag=wt[0]=0;
   tatavg=tat[0]=bt[0];
   for(i=1;i<n;i++)
      {
         wt[i]=wt[i-1]+bt[i-1];
         tat[i]=tat[i-1]+bt[i];
         wtag=wtag+wt[i];
         tatavg=tatavg+tat[i];
```

```
     }
   printf("PROCESS\t\t SYSTEM/USER PROCESS \tBURST TIME\tWAITING\tTIME\tTURNAROUND TI
ME\n");
   for(i=0;i<n;i++)
   printf("%d \t\t %d \t\t %d \t\t %d \t\t %d \n",p[i],su[i],bt[i],wt[i],tat[i]);
   printf("Average Waiting Time is: %f\n",wtag/n);
   printf("Average Turnaround Time is: %f\n",tatavg/n);
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the number of processes:  4
Enter the Burst Time of Process 0:  3
System/User Process (0/1):  1
Enter the Burst Time of Process 1:  2
System/User Process (0/1):  0
Enter the Burst Time of Process 2:  5
System/User Process (0/1):  1
Enter the Burst Time of Process 3:  1
System/User Process (0/1):  0
PROCESS          SYSTEM/USER PROCESS    BURST TIME      WAITING TIME    TURNAROUND TIME
1                0                2                0                2
3                0                1                2                3
2                1                5                3                8
0                1                3                8                11
Average Waiting Time is: 3.250000
Average Turnaround Time is: 6.000000
```

Test Case - 2

User Output

```
Enter the number of processes:  4
Enter the Burst Time of Process 0:  6
System/User Process (0/1):  1
Enter the Burst Time of Process 1:  9
System/User Process (0/1):  0
Enter the Burst Time of Process 2:  7
System/User Process (0/1):  1
Enter the Burst Time of Process 3:  5
System/User Process (0/1):  0
PROCESS          SYSTEM/USER PROCESS    BURST TIME      WAITING TIME    TURNAROUND TIME
1                0                9                0                9
3                0                5                9                14
2                1                7                14               21
0                1                6                21               27
Average Waiting Time is: 11.000000
Average Turnaround Time is: 17.750000
```