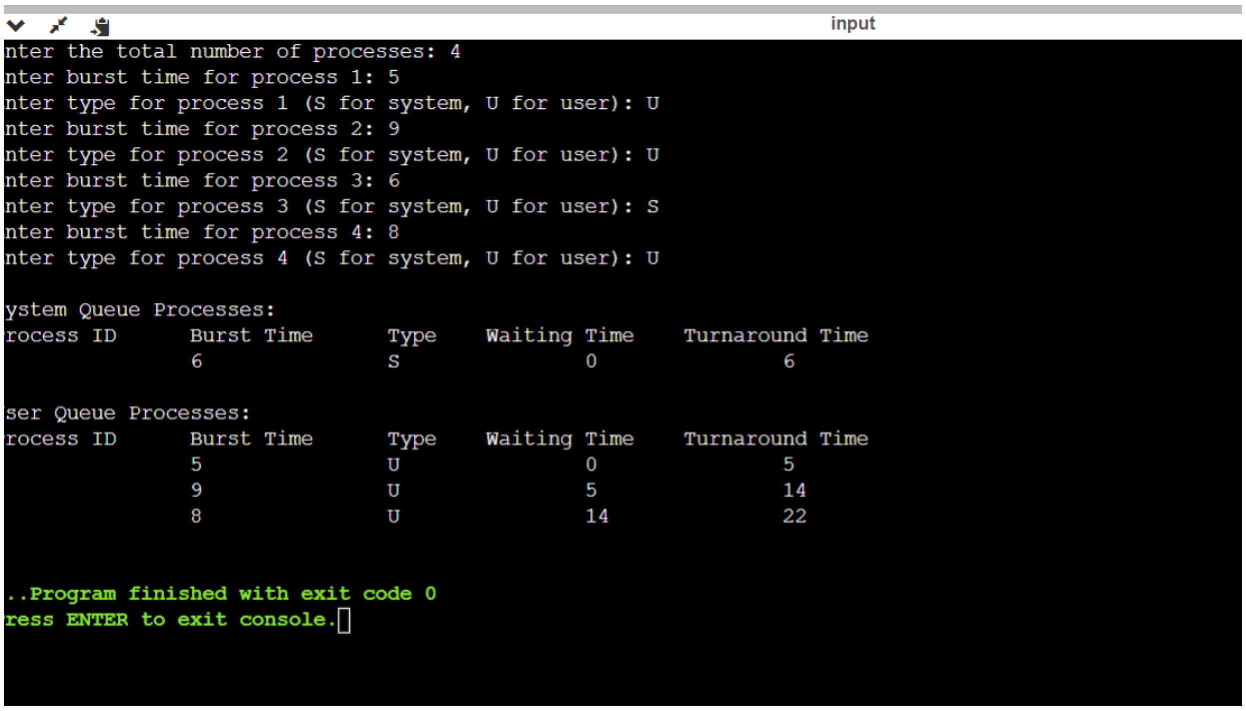
Assignment 4

Write a C program to simulate a multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.

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

typedef struct process {
   int id;
   int burstTime;
   char type; // 'S' for system process, 'U' for user process
   int waitingTime;
   int turnaroundTime;
} Process;

void calculateTime(Process queue[], int count) {
   int time = 0;
   for (int i = 0; i < count; i++) {
      queue[i].waitingTime = time;
      queue[i].turnaroundTime = time + queue[i].burstTime;
      time += queue[i].burstTime;
   }
}

void printQueue(Process queue[], int count) {
   printf("Process ID\tBurst Time\tType\tWaiting Time\tTurnaround Time\n");
   for (int i = 0; i < count; i++) {
      printf("%d\t\t%d\t\t%c\t\t%d\t\t%d\n", queue[i].id, queue[i].burstTime,
queue[i].type, queue[i].waitingTime, queue[i].turnaroundTime);
   }
}

int main() {
   int n, systemCount = 0, userCount = 0;

   printf("Enter the total number of processes: ");
   scanf("%d", &n);

   Process processes[n], *systemQueue, *userQueue;
```

```
    for (int i = 0; i < n; i++) {
      processes[i].id = i + 1;
      printf("Enter burst time for process %d: ", i + 1);
      scanf("%d", &processes[i].burstTime);
      do {
         printf("Enter type for process %d (S for system, U for user): ", i + 1);
         scanf(" %c", &processes[i].type); // Notice the space before %c to catch any
leftover whitespace
      } while (processes[i].type != 'S' && processes[i].type != 'U' && processes[i].type !=
's' && processes[i].type != 'u');

      if (processes[i].type == 'S' || processes[i].type == 's') {
         systemCount++;
      } else {
         userCount++;
      }
   }

   systemQueue = (Process *)malloc(systemCount * sizeof(Process));
   userQueue = (Process *)malloc(userCount * sizeof(Process));

   int sysIndex = 0, userIndex = 0;
     for (int i = 0; i < n; i++) {
      if (processes[i].type == 'S' || processes[i].type == 's') {
         systemQueue[sysIndex++] = processes[i];
      } else {
         userQueue[userIndex++] = processes[i];
      }
   }

    calculateTime(systemQueue, systemCount);
    calculateTime(userQueue, userCount);

   printf("\nSystem Queue Processes:\n");
   printQueue(systemQueue, systemCount);

   printf("\nUser Queue Processes:\n");
   printQueue(userQueue, userCount);

   free(systemQueue);
   free(userQueue);
   return 0;
}
```



```
nter the total number of processes: 4
nter burst time for process 1: 5
nter type for process 1 (S for system, U for user): U
nter burst time for process 2: 9
nter type for process 2 (S for system, U for user): U
nter burst time for process 3: 6
nter type for process 3 (S for system, U for user): S
nter burst time for process 4: 8
nter type for process 4 (S for system, U for user): U

ystem Queue Processes:
rocess ID      Burst Time      Type    Waiting Time    Turnaround Time
               6               S               0               6

ser Queue Processes:
rocess ID      Burst Time      Type    Waiting Time    Turnaround Time
               5               U               0               5
               9               U               5               14
               8               U               14              22


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