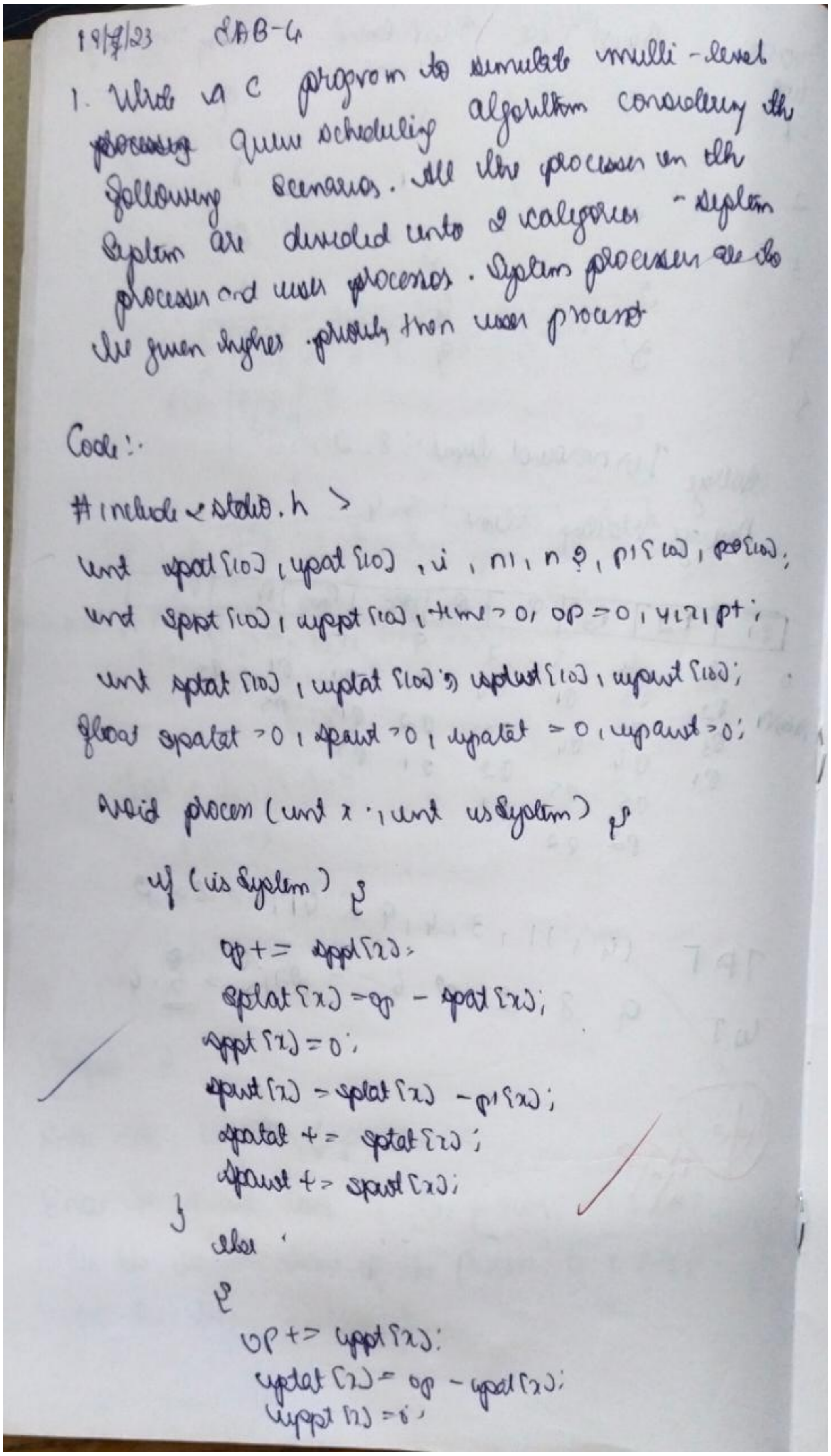
OS LAB-4

Q) Write a C program to simulate Real-Time CPU Scheduling algorithms: a)Multilevel queue scheduling b) Rate- Monotonic

CODE-



19/7/23 LAB-4

1. Write a C program to simulate multi-level queue scheduling algorithm considering the following scenarios. All the processes in the system are divided into 2 categories - system processes and user processes. System processes are to be given higher priority than user process

Code:-

```
#include<stdio.h>
int spat[10], upat[10], i, n1, n2, p1[10], p2[10];
int sppt[10], uppt[10], time=0, op=0, x, z, pt;
int sptat[10], uptat[10], upwt[10], upwt[10];
float spatat=0, spawt=0, upatat=0, upawt=0;

void process(int x, int isSystem) {

    if (isSystem) {
        op += sppt[x];
        sptat[x] = op - spat[x];
        sppt[x] = 0;
        spwt[x] = sptat[x] - p1[x];
        spatat += sptat[x];
        spawt += spwt[x];
    }
    else
    {
        op += uppt[x];
        uptat[x] = op - upat[x];
        uppt[x] = 0;
        upwt[x] = uptat[x] - pa[x];
        upatat += uptat[x];
        upawt += upwt[x];
    }
}
int main() {
        printf("Enter the no of System Processes: ");
        scanf("%d", &n1);
    printf("Enter the no of user processes : \n");
    ~~forkaren~~ scanf("%d", &n2);
    printf(" Enter the arrival times for System Processes:");
    for(i=0; i<n1; i++)
        scanf("%d", &spat[i]);
    printf("Enter the burst time for System processes: \n")
        for(i=0; i<n1; i++)
            scanf("%d", &sppt[i]);
        printf("Enter the arrival and burst time for
            user processor");
        for(i=0; i<n2; i++) {
            scanf("%d", &upat[i]);
            scanf("%d", &uppt[i]);
        }
        for(i=0; i<n1; i++) {
            time += sppt[i];
            pt[i] = sppt[i];
        }
```

```
for(i=0; i<n2; i++){

    time+= uppd[i];
    pd[i]= uppd[i];
    }
printf("\n");
while(op<time){

    y=-1;
    z=-1;
    ~~}~~
for(i=0; i<n1; i++){
    if(op>=spat[i] && sppt[i]!=0)
    {
      y=i;
      break;
  }
}
for(i=0; i<n2; i++){
      if(op>=upat[i] && uppt[i]!=0){
          z=i;
          break;
      }}

    if(y!=-1){
printf("%d SP%d", op, y+1);
    sproc(y,1);
  } else if(z!=-1){
```

```
printf ("%d UP%d", op, z+1);
        procen (z, 0);
      } else {
           op ++;
         }
      }
      printf ("%d", op);
   printf ("\n");

printf ("System Processes: \n");
  for(i=0; i<n1; i++)
     printf ("SP%d %d %d\n", i+1, sptat[i]);
  printf ("Average Turnaround Time (System Processes):
        %.2f\n", sptat /n1);
 printf (" Average Waiting Time (System Processes): %.2f\n");
 ~~printf (" SP %d %d \n"~~ printf (" spwt\n);
 printf (" \n");

printf ("User Processes: \n");
 for(i=0; i<n2; i++)
 printf ("UP%d %d %d \n", i+1, uptat[i], upwt[i]);
printf ("Average Turnaround Time(User Processes) : %.2f \n",
    uptat /n2);
 printf (" Average Waiting Time (User Processes ): %.2f\n",
   upwt /n2);
   return 0;
 }
```

19/9/23 Lab-6.

1. Write a C program to simulate Real time Scheduling algorithm

i) Rate monotonic

ii) Earliest deadline first

iii) Proportional scheduling

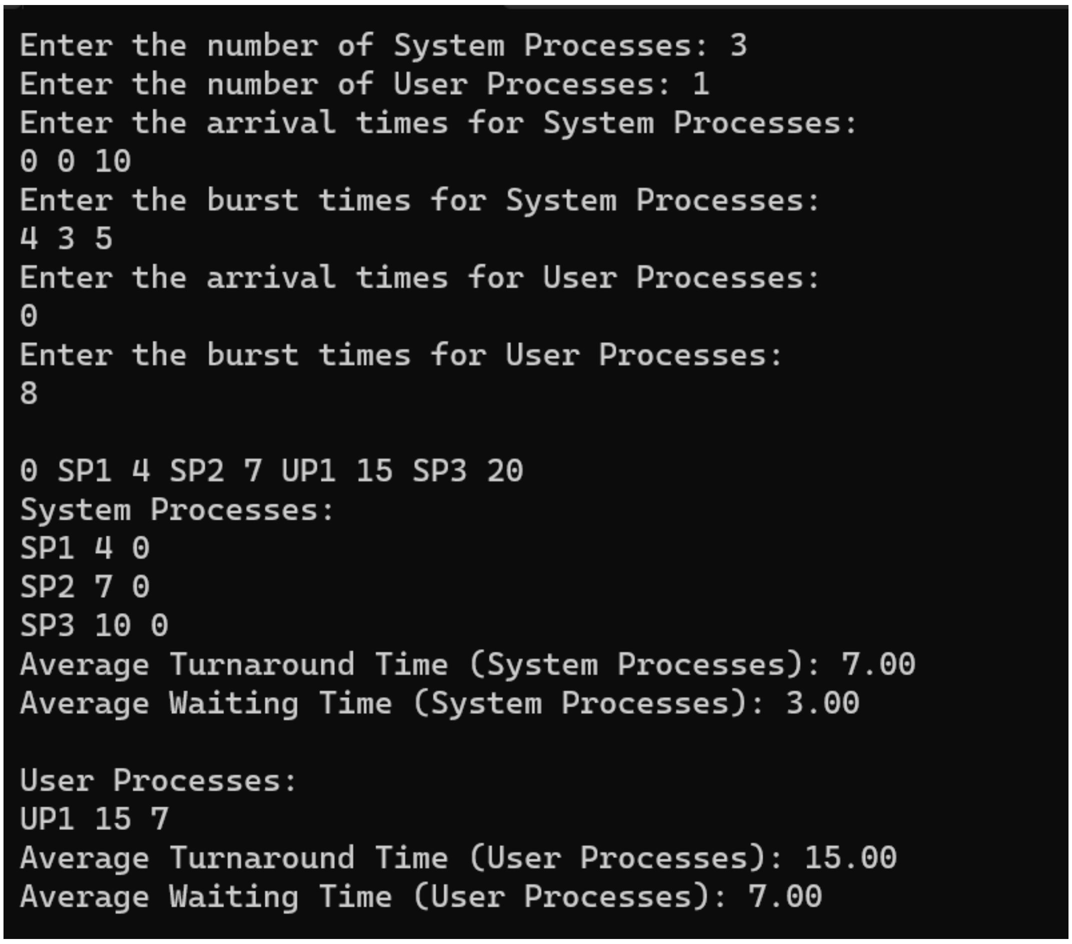
```
#include <stdio.h>
#include <stdlib.h>
int bt[20], n, i, d[20], p[10], re[10], flag=1;
int main()
{
  int ch, k=1;
  while(k)
  {
  printf("1. Monotone 2. Earliest Deadline 3.
         Proportional 4. End ");
  printf(" Enter your choice: ");
   scanf("%d", &ch);
   printf("Enter no of processes");
    scanf("%d", &n);
   printf(" Enter execution & deadline");
     for(int i=0; i<n; i++)
     {
       scanf("%d", &bt[i]);
      scanf(" %d", &d[i]);
     }
```

```
for(i=0; i<n; i++)
  {    p[i] = bt[i]
       re[i] = 0;
   }

 switch(ch)
{
 case 1: mono();
         break;

 case 2: eaf();
         break;

 ~~case 3: prop();~~
         ~~break;~~

 case 3: exit(0);
         break;
  default: printf("invalid choice");
}
 int lcm(int a, int b)
 {
   int max = (a>b)? a : b;
 while(1)
 {
  if (max % a == 0 && max % b == 0)
      return max;
    }
 }
int lcmArr(int a[], int n).
   {   int res = a[0];
   for( int i=1; i<n; i++)
       res = lcm(result, a[i]);
    return result;
  }

void mono ()
  {
    int t = len(d, n)
    int op = 0, pr=0, pre= pr;
    while (op <= t)
      {
        for( i=0; i<n; i++).
    {
     if (op % d[i]==0)
        ~~ready~~ re [i] =1;
      }
      flg=0;
      for( i = 0; i<n; i++)
      {
       if ( re[i] == 1)
           flg =1;
      }
       if ( flg ==0)
          pr = -1;
       else.
        {
          pr =-1;
           for( i=0; i<n; i++)
               {.
```

```
if (re[i] == 1)
  {
    if (pr == -1 || d[i] < d[pr])
              pr = [i];

 }}}

  if (pr != pre)
    {
      if (pr == -1)
        printf (" %idle " ,op);
      else
      printf ("%d %d %d (op ,pr+1);
     op++;
   if (pr != -1)
    {
     p[pr] = p[pr] - 1;
      if (p[pr] == 0)
     {
      p[pr] = et(pr);
      re[pr] = 0;
     }}
      pre = pr;

   }
 }

void ulf() {
    int time = len(d, n);
    int op=0, pr=0, pre=-1, flag, i;
          while(op <= time) {
```

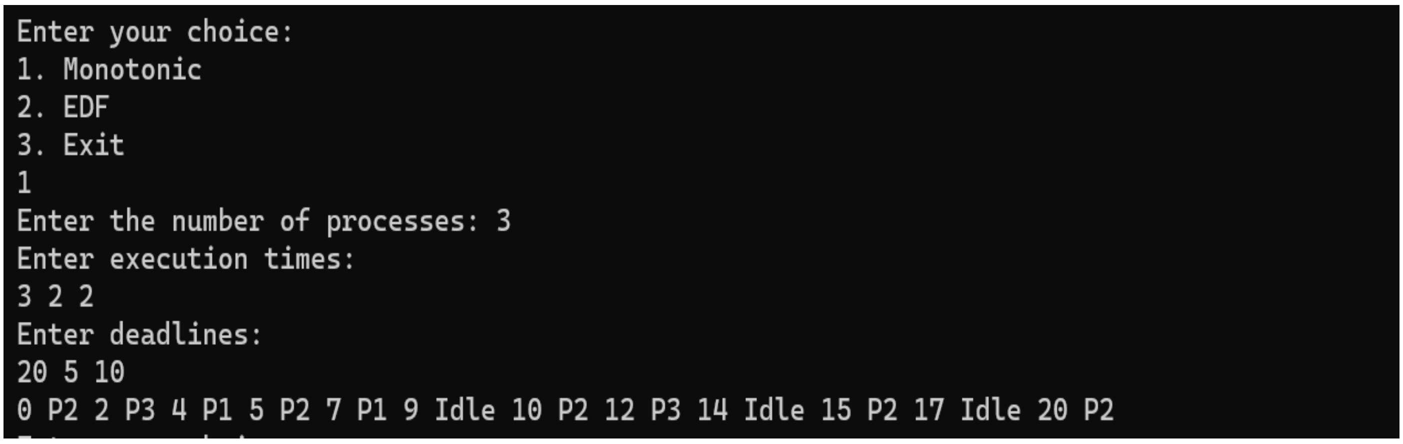
OUTPUT-

Multilevel queue Scheduling



```
Enter the number of System Processes: 3
Enter the number of User Processes: 1
Enter the arrival times for System Processes:
0 0 10
Enter the burst times for System Processes:
4 3 5
Enter the arrival times for User Processes:
0
Enter the burst times for User Processes:
8

0 SP1 4 SP2 7 UP1 15 SP3 20
System Processes:
SP1 4 0
SP2 7 0
SP3 10 0
Average Turnaround Time (System Processes): 7.00
Average Waiting Time (System Processes): 3.00

User Processes:
UP1 15 7
Average Turnaround Time (User Processes): 15.00
Average Waiting Time (User Processes): 7.00
```

Rate Monotonic



```
Enter your choice:
1. Monotonic
2. EDF
3. Exit
1
Enter the number of processes: 3
Enter execution times:
3 2 2
Enter deadlines:
20 5 10
0 P2 2 P3 4 P1 5 P2 7 P1 9 Idle 10 P2 12 P3 14 Idle 15 P2 17 Idle 20 P2
```