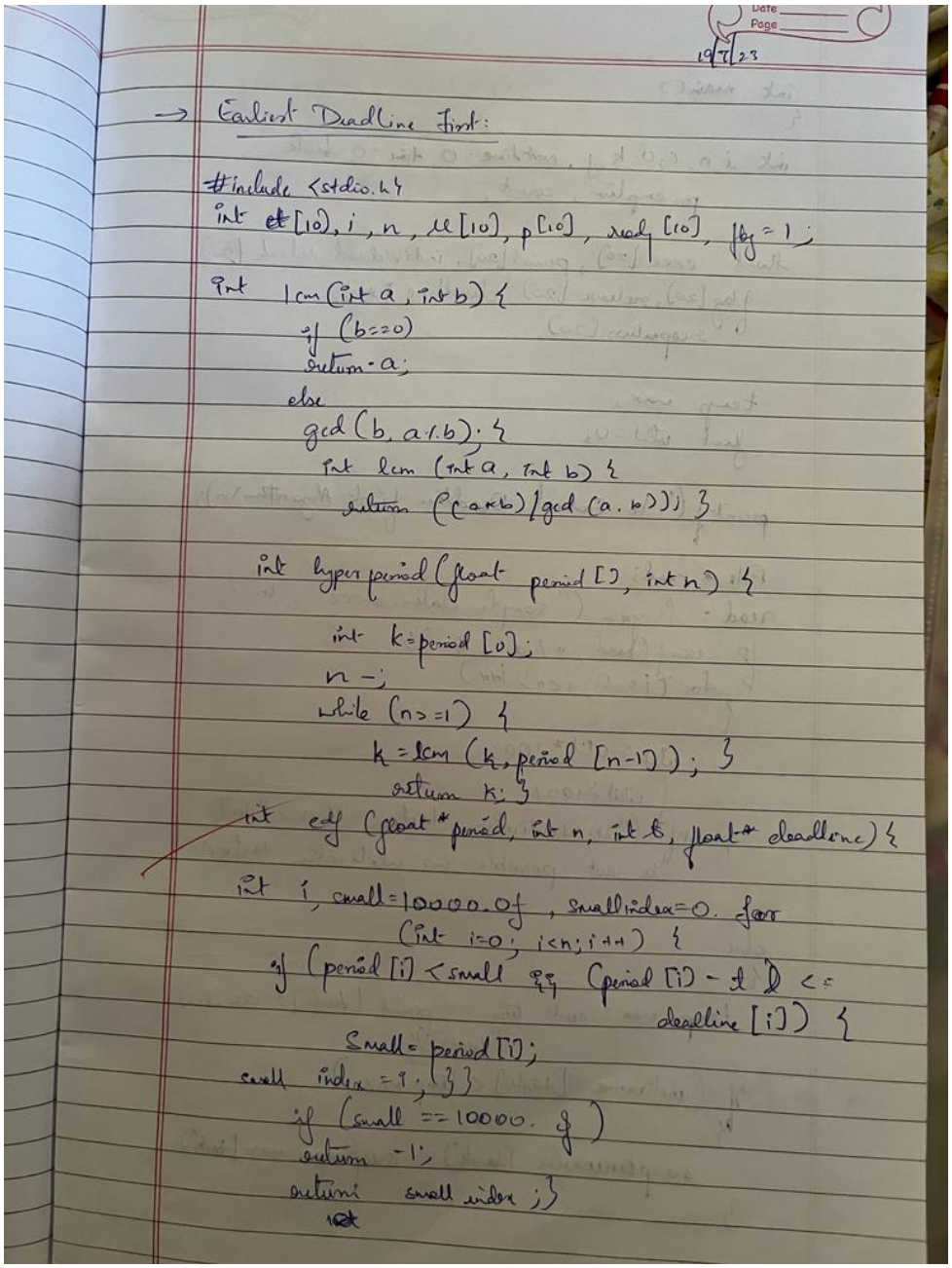
WEEK4

Write a C program to simulate Real-Time CPU Scheduling algorithms:Earliest-deadline First



→ Earliest Deadline First:

```
#include <stdio.h>
int et[10], i, n, d[10], p[10], rem[10], flag=1;
int lcm(int a, int b) {
    if (b==0)
    return a;
    else
    gcd(b, a%b); }
    int lcm (int a, int b) {
    return ((a*b)/gcd(a, b))); }

int hyperperiod(float period[], int n) {
    int k=period[0];
    n--;
    while (n>=1) {
        k=lcm(k, period[n-1]); }
        return k; }
int edf(float *period, int n, int t, float *deadline) {
int i, small=10000.0f, smallindex=0, for
        (int i=0; i<n; i++) {
if (period[i]<small && (period[i]-t)<=
                                deadline[i]) {
        small=period[i];
small index=i; }}
    if (small==10000.0f)
    return -1;
    return small index; }
```

```
int main()
{
int i, n, c, d, k, j, nextdime = 0, time = 0, task,
    preemption, count;

float exec[20], period[20], individual_util[20]
  flag[20], release[20], deadline[20],
    response[20],

  temp max;
  float util = 0;

printf("\n Earliest Deadline first Algorithm\n");
File * read;
read = fopen("sample data.docx", "r");
fscanf(read, "%d", &n);
for(i = 0; i<n; i++)
  {
    util = util * 100;
    if util >100)
printf("\n utilization factor = %.0.2f \n",
    is not possible as utilization factor);

else
{
  temp max = next time - (period[task] - deadline
                          [task]);
if (instance[task] < temp max)
{
  response[task] = response max[task];
}
```

```
preemption_count = 0;
for (i=0; i<k; i=j)
{
    flag[i] = 1;
    d = ex[i];
    for (j=i; d == ex[j]; j++)
    flag[d]++;
    if (flag[d] == exer[d])
}
printf ("\n\n Preemption count = %d",
        preemption_count);
}
return 0;
}
```

Output:

Enter your choice:
1. Monotonic
2. EDF

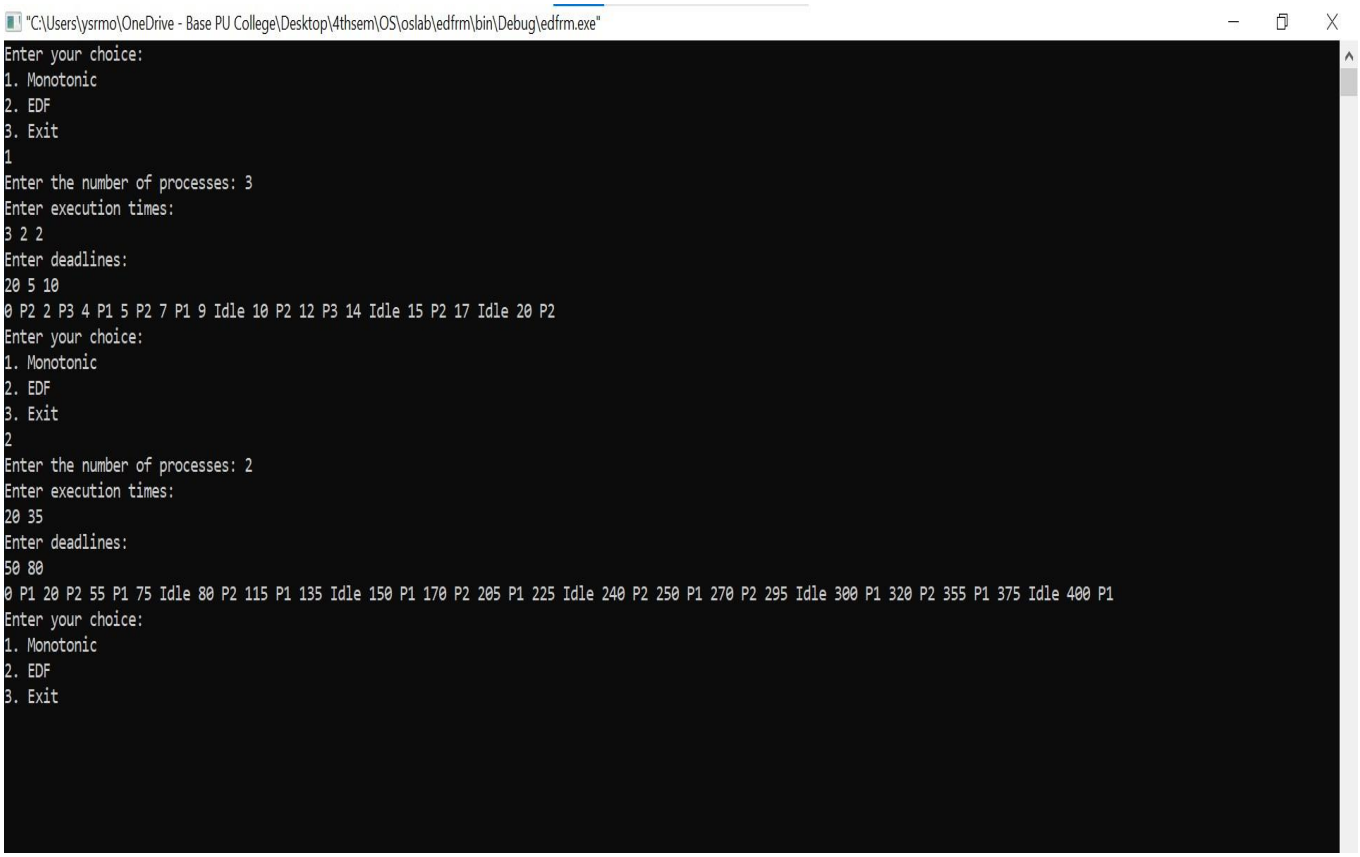
Enter number of processes: 3

Enter Execution time & deadline:

4	1
5	5
2	4

0 P_3 2 P_1 6 P_3 10 P_1 14 P_3 18 P_1

OUTPUT:


"C:\Users\ysrmo\OneDrive - Base PU College\Desktop\4thsem\OS\oslab\edfrm\bin\Debug\edfrm.exe"
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
Enter your choice:
1. Monotonic
2. EDF
3. Exit
2
Enter the number of processes: 2
Enter execution times:
20 35
Enter deadlines:
50 80
0 P1 20 P2 55 P1 75 Idle 80 P2 115 P1 135 Idle 150 P1 170 P2 205 P1 225 Idle 240 P2 250 P1 270 P2 295 Idle 300 P1 320 P2 355 P1 375 Idle 400 P1
Enter your choice:
1. Monotonic
2. EDF
3. Exit