WEEK 4

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

CODE:

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


int et[10], i, n, dl[10], p[10], ready[10], flag = 1;

int lcm(int a, int b) {
    int max = (a > b) ? a : b;
    while (1) {
        if (max % a == 0 && max % b == 0)
            return max;
        max++;
    }
}

int lcmArray(int arr[], int n) {
    int result = arr[0];
    for (int i = 1; i < n; i++) {
        result = lcm(result, arr[i]);
    }
    return result;
}

void mono() {
    int time = lcmArray(dl, n);
    int op = 0, pr = 0, pre = pr;



    while (op <= time) {
        for (i = 0; i < n; i++) {
```

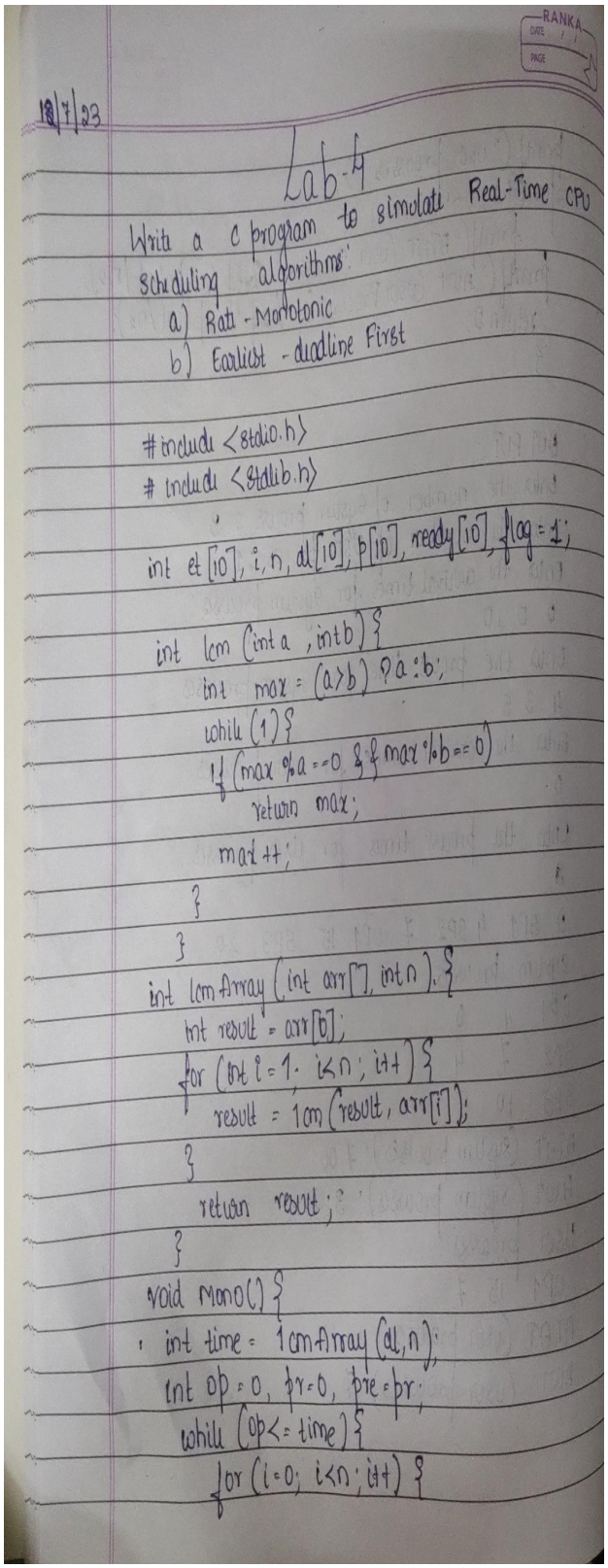
```
    if (op % dl[i] == 0) {
        ready[i] = 1;
    }
}

flag = 0;
for (i = 0; i < n; i++) {
    if (ready[i] == 1) {
        flag = 1;
        break;
    }
}

if (flag == 0) {
    pr = -1;
} else {
    pr = -1;
    for (i = 0; i < n; i++) {
        if (ready[i] == 1) {
            if (pr == -1 || dl[i] < dl[pr]) {
                pr = i;
            }
        }
    }
}



    if (pr != pre) {
        if (pr == -1) {
            printf("%d Idle ",op);
        } else {
            printf("%d P%d ",op, pr + 1);
        }
    }

op++;
if (pr != -1) {
    p[pr] = p[pr] - 1;
    if (p[pr] == 0) {
        p[pr] = et[pr];
```

```
            ready[pr] = 0;
        }
    }

    pre = pr;
  }

  printf("\n");
}



void edf() {
  int time = lcmArray(dl, n);
  int op = 0, pr = 0, pre = -1;
  int flag, i;

  while (op <= time) {
    for (i = 0; i < n; i++) {
      if (op % dl[i] == 0) {
        ready[i] = 1;
      }
    }

    flag = 0;
    for (i = 0; i < n; i++) {
      if (ready[i] == 1) {
        flag = 1;
        break;
      }
    }

    if (flag == 0) {
      pr = -1;
    } else {
      pr = -1;
      for (i = 0; i < n; i++) {
        if (ready[i] == 1) {
          if (pr == -1 || p[i] < p[pr]) {
            pr = i;
```

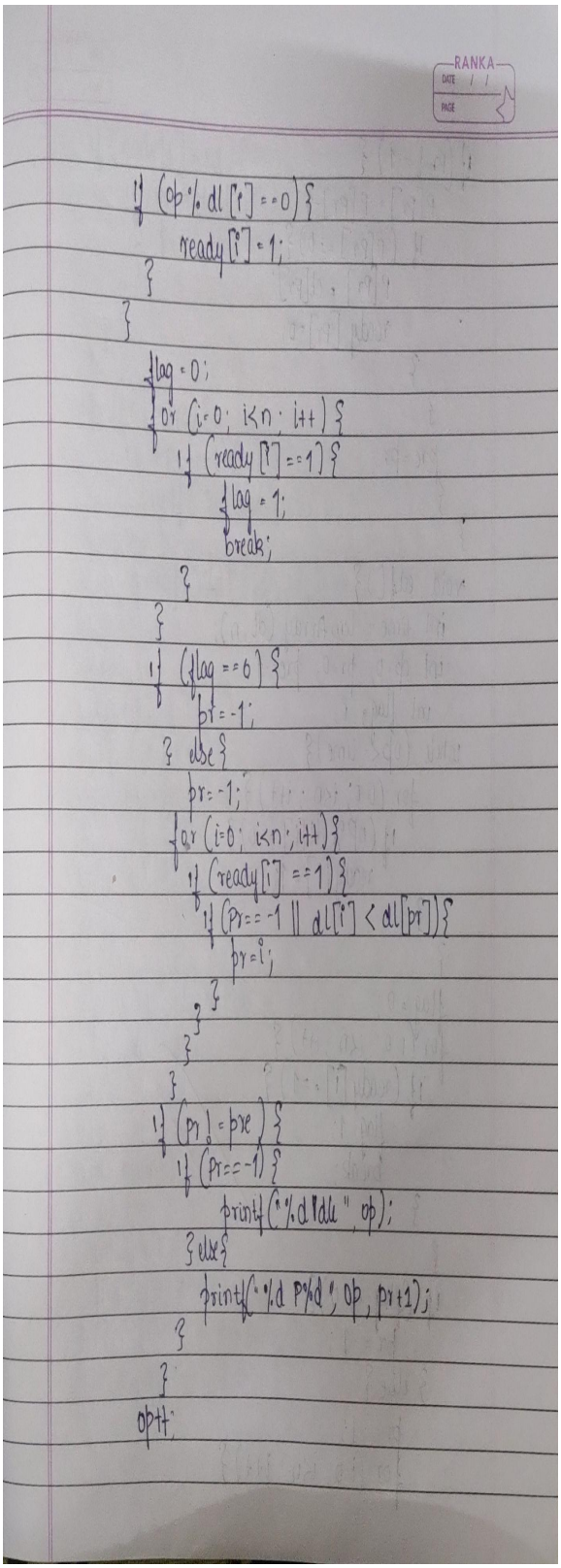
```
                }
            }
        }
    }

    if (pr != pre) {
        if (pr == -1) {
            printf("%d Idle ", op);
        } else {
            printf("%d P%d ", op, pr + 1);
        }
    }

    op++;

    if (pr != -1) {
        p[pr] = p[pr] - 1;
        if (p[pr] == 0) {
            p[pr] = et[pr];
            ready[pr] = 0;
        }
    }

    pre = pr;
  }

  printf("\n");
}

int main() {
  int ch, k = 1;
  while (k) {
    printf("Enter your choice: \n1. Monotonic \n2. EDF \n3. Proportional \n4. Exit\n");
    scanf("%d", &ch);
    if(ch==3)
    exit(0);
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter execution times: \n");
    for (i = 0; i < n; i++)
```

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

        printf("Enter deadlines: \n");
        for (i = 0; i < n; i++)
            scanf("%d", &dl[i]);

        for (i = 0; i < n; i++)
            p[i] = et[i];

        for (i = 0; i < n; i++)
            ready[i] = 0;

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

            case 2:
                edf();
                break;

            case 3:
                k = 0;
                break;

            default:
                printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

OBSERVATION:



```
19/7/23

Lab-4

Write a C program to simulate Real-Time CPU
scheduling algorithms:
a) Rate-Monotonic
b) Earliest-deadline First

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

int et[10], i, n, dl[10], p[10], ready[10], flag=1;

int lcm(int a, int b){
    int max = (a>b) ?a:b;
    while(1){
    if(max%a==0 && max%b==0)
        return max;
    max++;
    }
}
int lcmArray(int arr[], int n){
    int result = arr[0];
    for(int i=1; i<n; i++){
        result = lcm(result, arr[i]);
    }
    return result;
}
void mono(){
    int time = lcmArray(dl,n);
    int op=0, pr=0, pre=pr;
    while(op<=time){
        for(i=0; i<n; i++){
```



```
            if(op%dl[i]==0){
                ready[i]=1;
            }
        }
        flag=0;
        for(i=0; i<n; i++){
            if(ready[i]==1){
                flag=1;
                break;
            }
        }
        if(flag==0){
            pr=-1;
        }else{
            pr=-1;
            for(i=0; i<n; i++){
                if(ready[i]==1){
                    if(pr==-1 || dl[i]<dl[pr]){
                        pr=i;
                    }
                }
            }
        }
        if(pr!=pre){
            if(pr==-1){
                printf("%d idle", op);
            }else{
                printf("%d P%d", op, pr+1);
            }
        }
    }
    op++;
```

```
if (pr != -1) {
    P[pr] = P[pr] - 1;
    if (P[pr] == 0) {
        P[pr] = et[pr];
        ready[pr] = 0;
    }
}
pre = pr;
}
}

void edf() {
int time = lcmArray(dl, n);
int op = 0, pr = 0, pre = -1;
int flag, i;
while (op <= time) {
    for (i = 0; i < n; i++) {
        if (op % dl[i] == 0) {
            ready[i] = 1;
        }
    }
    flag = 0;
    for (i = 0; i < n; i++) {
        if (ready[i] == 1) {
            flag = 1;
            break;
        }
    }
    if (flag == 0) {
        pr = -1;
    } else {
        pr = -1;
        for (i = 0; i < n; i++) {
```

```
if (ready[i] == 1) {
    if (pr == -1 || P[i] < P[pr]) {
        pr = i;
    }
}
}
if (pr != pre) {
    if (pr == -1) {
        printf("%d idle", op);
    } else {
        printf("%d P%d", op, pr+1);
    }
}
op++;
if (pr != -1) {
    P[pr] = P[pr] - 1;
    if (P[pr] == 0) {
        P[pr] = et[pr];
        ready[pr] = 0;
    }
}
pre = pr;
}
}

int main() {
int ch, k = 1;
while(k) {
printf("Enter your choice: \n1. Monotonic \n2. EDF \n
    3. proportional \n4. Exit \n");
scanf("%d", &ch);
if (ch == 3)
exit(0);
```

```
printf("Enter number of process:");
scanf("%d", &n);
printf("Enter execution time:\n");
    for(i=0; i<n; i++)
      scanf("%d", &et[i]);
  printf("Enter deadline:\n");
    for(i=0; i<n; i++)
      scanf("%d", &dl[i]);
  for(i=0; i<n; i++)
    p[i] = et[i];
  for(i=0; i<n; i++)
    ready[i] = 0;
  switch(ch)
  {
   case 1: mono();
           break;
   case 2: edf();
           break;
   case 3: k=0;
           break;
   default: printf("Invalid\n");
  }
 }
 return 0;
}
```

OUTPUT:

Enter your choice:
1. Monotonic
2. EDF
3. Exit
1

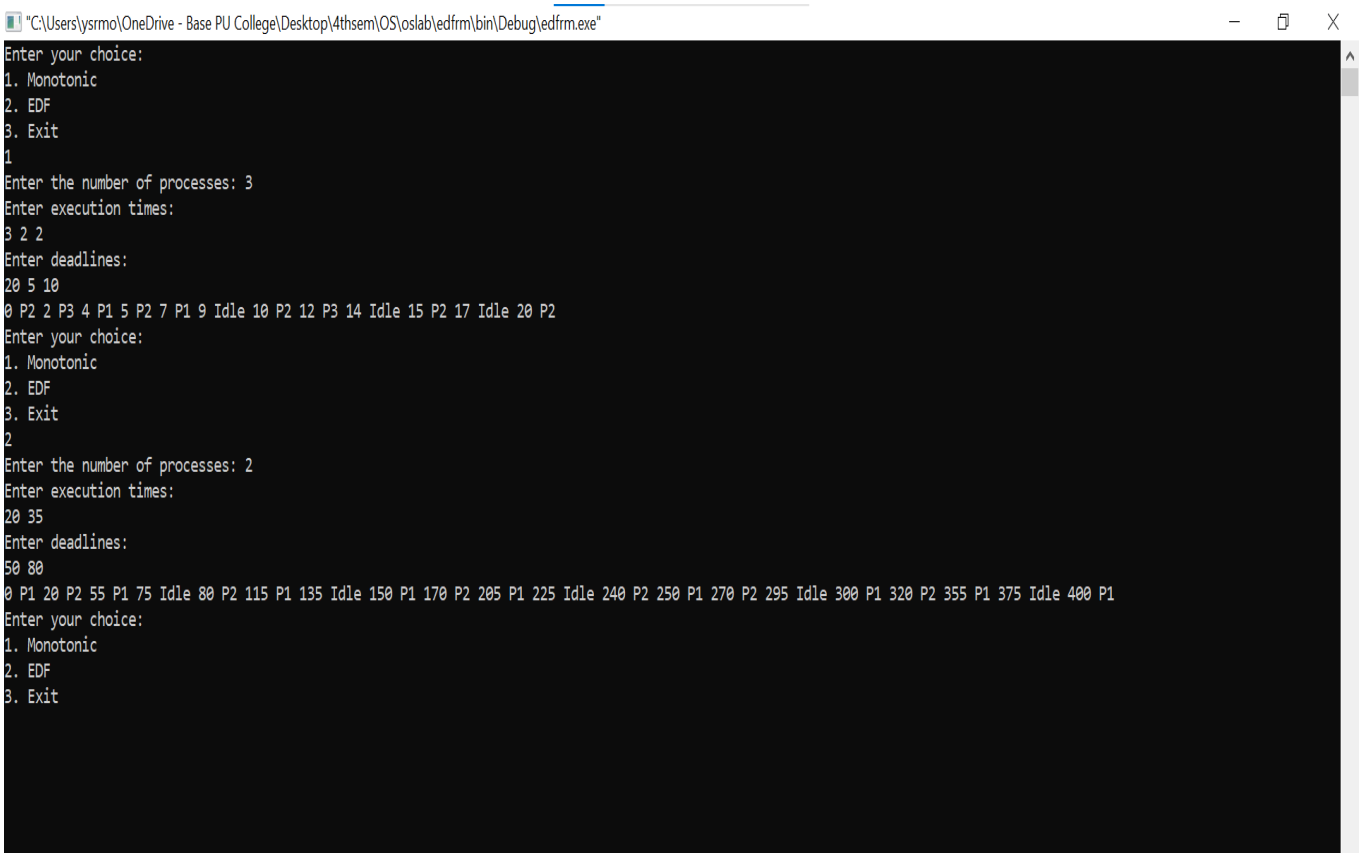
Enter the Number of process: 3
Enter Execution time:
3 2 2
Enter Deadlines:
20 5 10
0 P_2 2 P_3 4 P_1 5 P_3 7 P_1 9 Idle 10 P_2 12 P_3 14 Idle 15 P_2 17 Idle 20 P_2

Enter your choice:
1. Monotonic
2. EDF
3. Exit
2.
Enter the Number of process: 2
Enter Execution time:
20 35
Enter deadline:
50 80
0 P_1 20 P_2 55 P_1 75 Idle 80 P_2 115 P_1 135 Idle 150 P_1 170 P_2 205 P_1 225 Idle 240..

14/7/23

10/10

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