```
}
```

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;


   printf("%d ",op);
    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("P%d %d ", pr + 1,op);
```

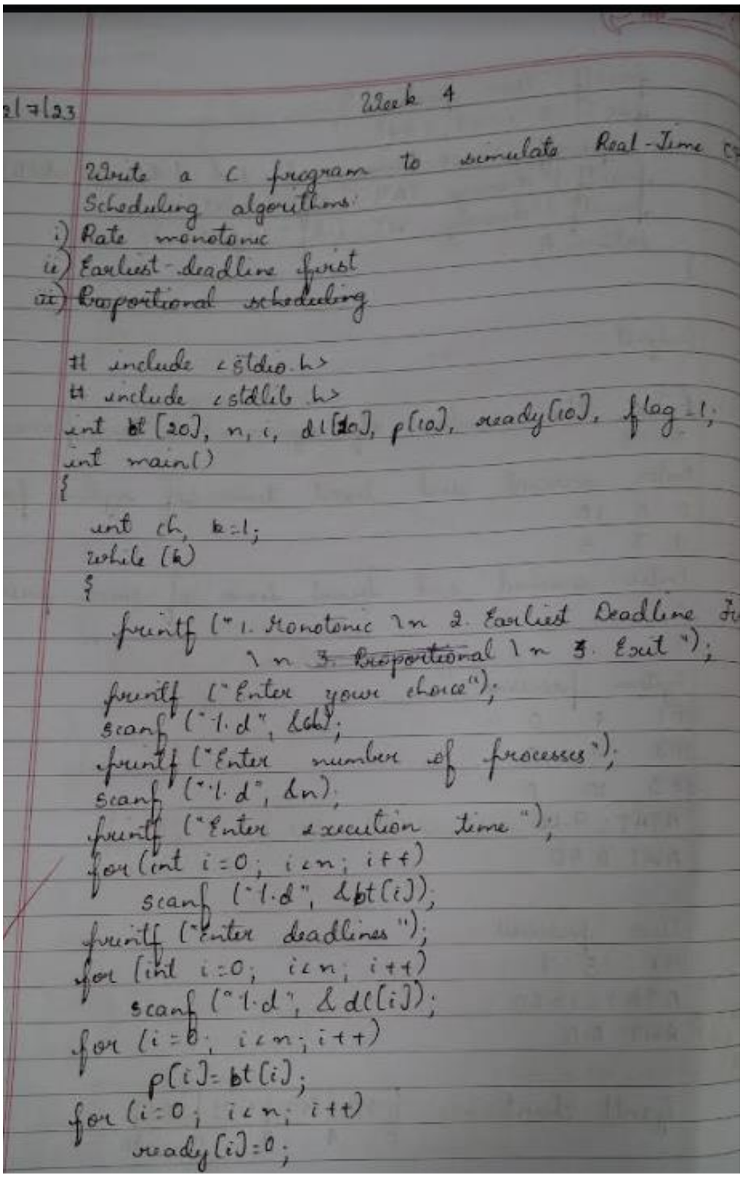
```
                }
            }

        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. Exit\n");
        scanf("%d", &ch);
        if(ch==4)
        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 book :



2|7|23 Week 4

Write a C program to simulate Real-Time C

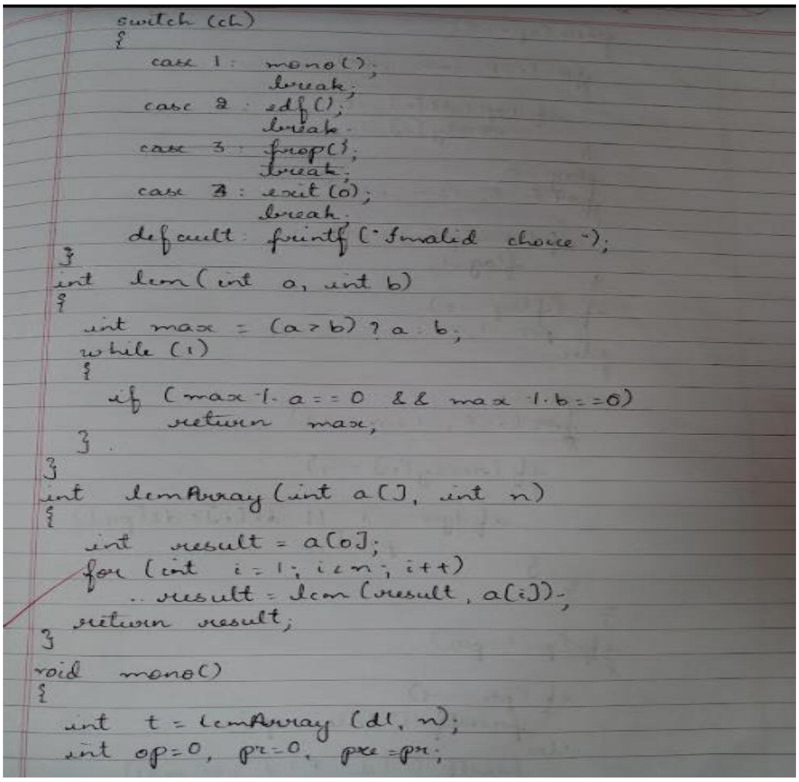
Scheduling algorithms:

i) Rate monotonic

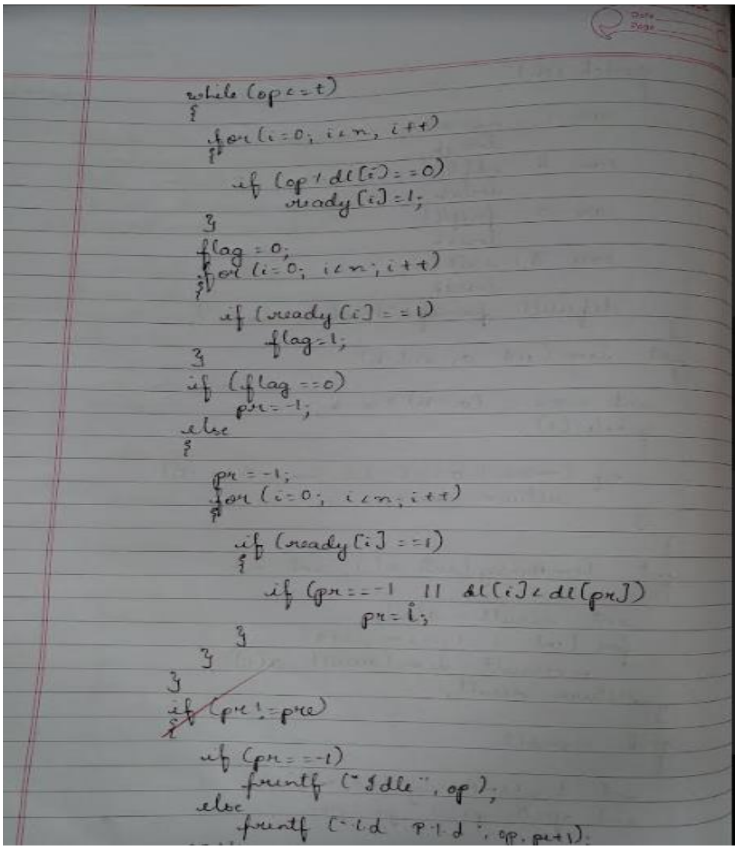
ii) Earliest-deadline first

iii) ~~Proportional scheduling~~

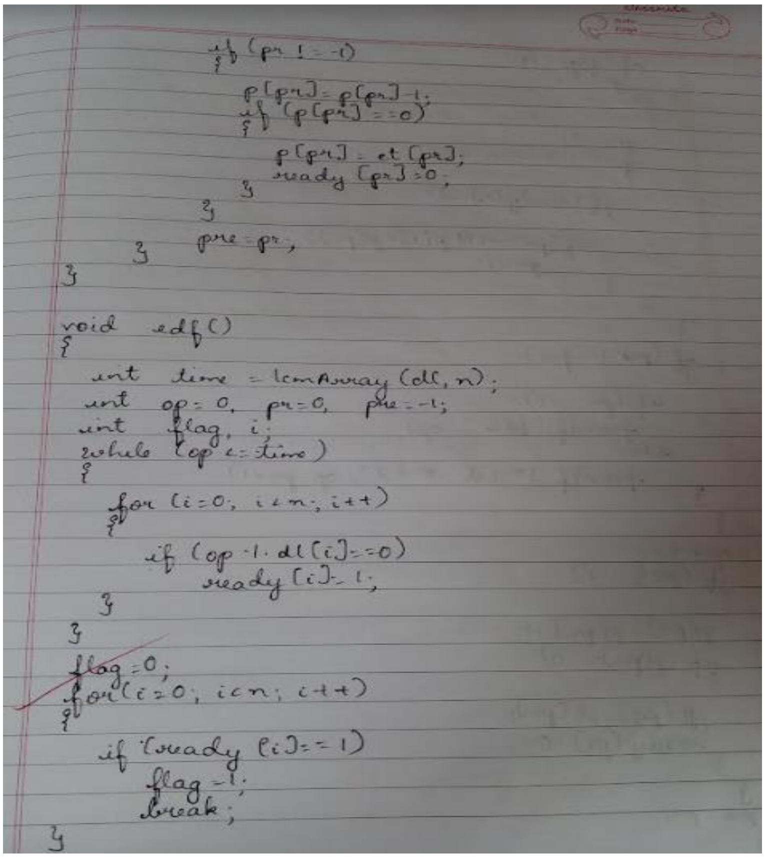
```
# include <stdio.h>
# include <stdlib.h>
int bt [20], n, i, d1[10], p[10], ready[10], flag=1;
int main()
{
  int ch, k=1;
  while (k)
  {
     printf ("1. Monotonic \n 2. Earliest Deadline Fi
              \n ~~3. Proportional~~ \n 3. Exit ");
    printf ("Enter your choice");
    scanf ("%d", &ch);
    printf ("Enter number of processes");
    scanf ("%d", &n);
    printf ("Enter execution time");
    for(int i=0; i<n; i++)
        scanf ("%d", &bt[i]);
   printf ("Enter deadlines");
   for (int i=0; i<n; i++)
       scanf ("%d", &d1[i]);
   for (i=0; i<n; i++)
        p[i]=bt[i];
   for (i=0; i<n; i++)
       ready[i]=0;
```



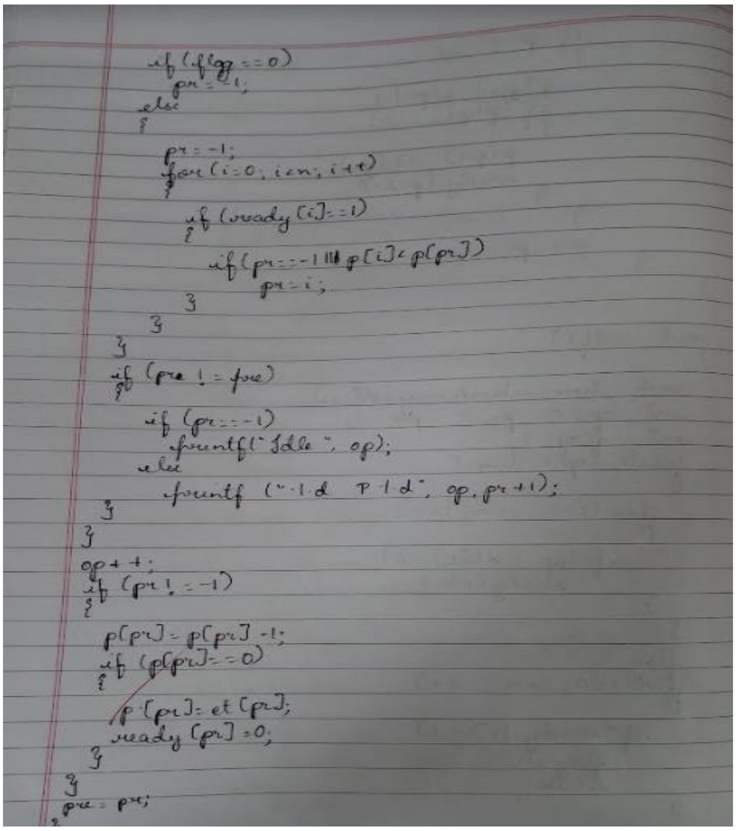
```
    switch (ch)
    {
        case 1 :  mono();
                  break;
        case 2 :  edf();
                  break.
        case 3 :  prop();
                  break;
        case 3 :  exit(0);
                  break.
       default :  printf("Invalid choice");
 }
int lcm(int a, int b)
{
   int max = (a>b) ? a : b;
   while (1)
   {
     if (max % a == 0 && max % b ==0)
         return max;
   }
}
int  lcmArray (int a[], int n)
{
   int result = a[0];
   for (int i = 1; i<n; i++)
      result = lcm (result, a[i]);
  return result;
}
void  mono()
{
  int t = lcmArray (dl, n);
  int op=0, pr=0, pre=pr;
```



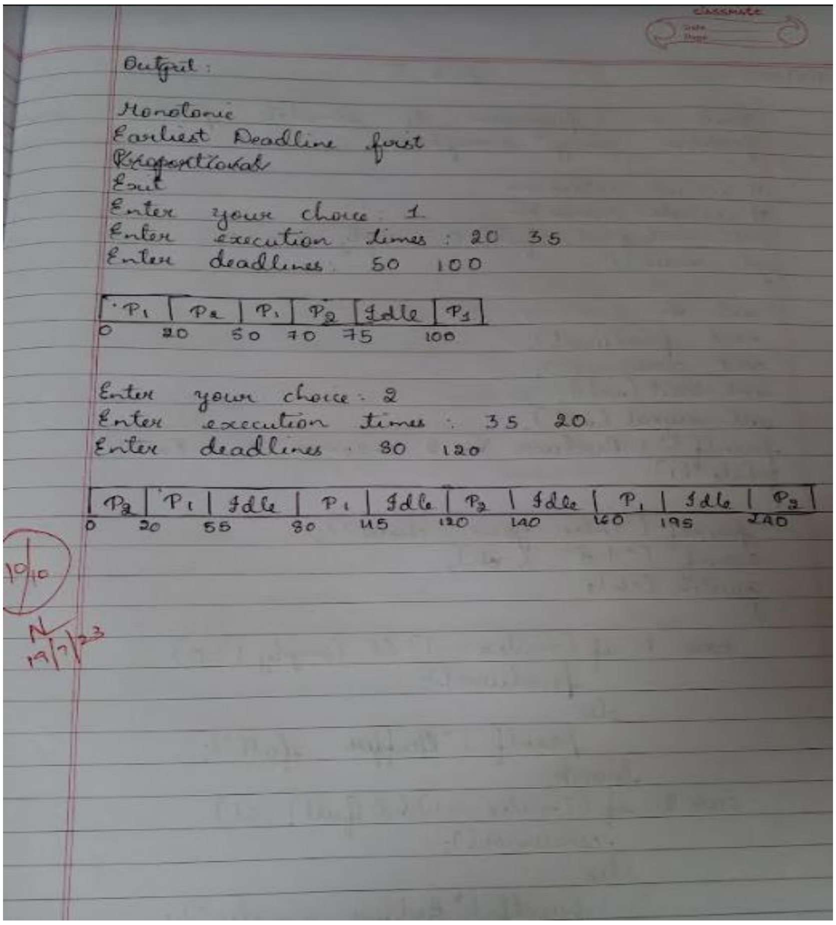
```
while (op<=t)
{
  for(i=0; i<n, i++)
  {
    if (op % dl[i]==0)
        ready[i]=1;
  }
 flag = 0;
 for (i=0; i<n; i++)
 {
   if (ready[i]==1)
       flag=1;
 }
if (flag==0)
   pr=-1;
else
{
  pr=-1;
  for (i=0; i<n; i++)
  {
    if (ready[i]==1)
    {
       if (pr==-1 || dl[i]<dl[pr])
        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



```
            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 (pre != pre)
      {
         if (pr == -1)
            printf("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;
}
```



Output :

Monotonic
Earliest Deadline first
~~Proportional~~
Exit
Enter your choice 1
Enter execution times : 20 35
Enter deadlines 50 100

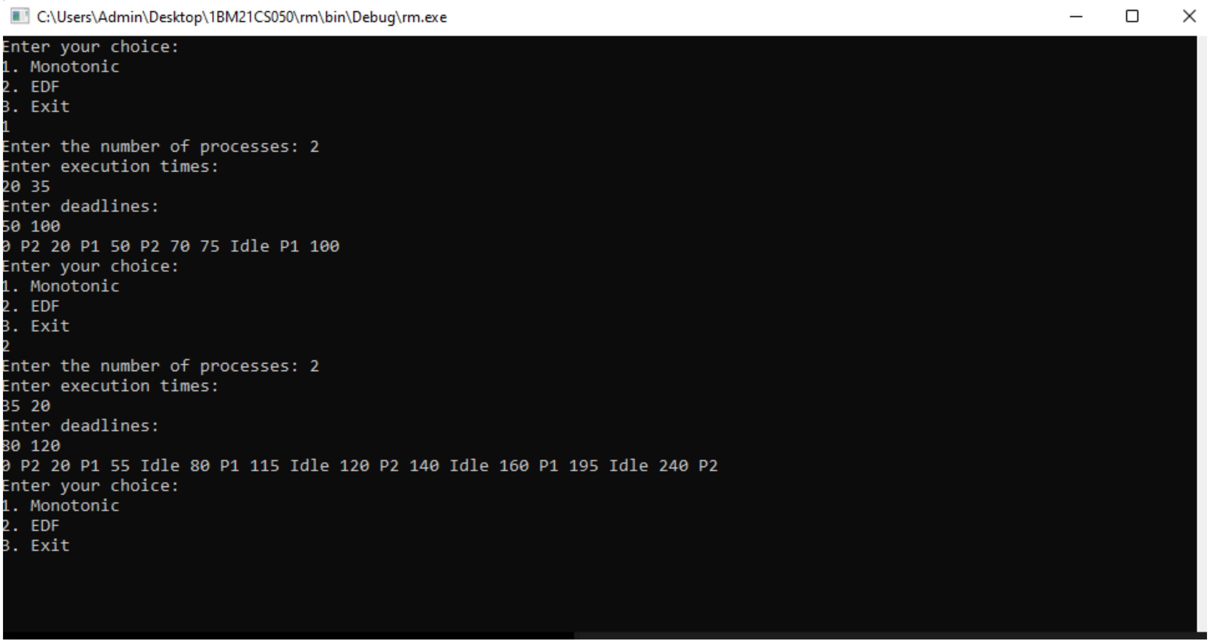
| P1 | P2 | P1 | P2 | Idle | P1 |
0 20 50 70 75 100

Enter your choice : 2
Enter execution times : 35 20
Enter deadlines : 80 120

| P2 | P1 | Idle | P1 | Idle | P2 | Idle | P1 | Idle | P2 |
0 20 55 80 115 120 140 160 195 240

10/10
N
19/7/23

Output :



```
Enter your choice:
1. Monotonic
2. EDF
3. Exit
1
Enter the number of processes: 2
Enter execution times:
20 35
Enter deadlines:
50 100
0 P2 20 P1 50 P2 70 75 Idle P1 100
Enter your choice:
1. Monotonic
2. EDF
3. Exit
2
Enter the number of processes: 2
Enter execution times:
35 20
Enter deadlines:
80 120
0 P2 20 P1 55 Idle 80 P1 115 Idle 120 P2 140 Idle 160 P1 195 Idle 240 P2
Enter your choice:
1. Monotonic
2. EDF
3. Exit
```