```
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 = IcmArray(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 % dI[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");
}
void prop() {
  // Implementation for proportional share scheduling algorithm
}
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==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:
          prop();
          break;
        case 4:
          k = 0:
          break;
        default:
          printf("Invalid choice.\n");
     }
  }
  return 0;
}
Output:
```

Rate Monotonic:

```
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
```

Earliest Deadline First:

```
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
```

Observation:

```
(C1) H
             13-02-23
              write a c program to simulate @ Rock Monotonie @ Familiest_
              deadline first. @ proportional Schuduling.
             @ #include Lstaio. 4)
                # Welude < Stdlib. 4>
                int et (10), i, n, d([10), P[10], ready[10), flag = 1;
                int lem (inta, int 6) of
                  jut mon = (asb) ? a : b;
                   while (1) of
                        i + (moun 0/0 a = = 0 to moun do b= =0)
                            return man;
                       mon ++ ;
               y
               but lemporary (lut aro [], but n) of
                    but result = orrs (0);
                  for liz1; icn; i+t) of
                       result = (congresult, arr (i)));
                   return result!
              int time = (comparay (dl, n);
                  white cope = time? of
                      for (1=0) i <n : 1++) d
                          it (opusalli) = = 01d
                            ready[i]=1;
                Plag = 0;
                  forci=0; icn; 1++) d
                       if ( sead(i) == 1) }
  8
                             May =1;
                            break :
                 ix ( Hay = = 0) 1
                        Pr = -1;
                I elsed
                    for (1:0) ( = n; i++) {
                      it (read (i) = = 17 f
                         1+ (pr==-111 dili) < di[pr])+
                             De = 1;
                 it ( pr != - pire ) d
 44 32-2014
                       it (m==-1) 1
                           print(" 90) Ille ", op);
 4) 3 bride
                           prints ( 1 4) of P 1/0d 1, OP 185+1);
```

```
7d (pr !=-1)d
                                                              94 (pr != pre) {
       1-[m]==[m] q
                                                              14 (00=-474
                                                                     print+ (" dod
       b (0==[19]9) +i
                                                                  gelsed
                                                                     private (a)
           b(be) = et(be);
          ready (M) =0; and complete for both
                                                               0P++;
                                                               1+ ( PE 1 = -1 ) {
   Dec = be!
                                                                   D[ 12] = P[1
 privet # (" (" ");
                                                                     [1997)
 3.
                                                                          PCP
  output:
                                                                           reo
 Enter number of processes: 3
                   tims '.
 Enter enecution
                                                               privat (" on");
  3 2 2
  Enter deadlines!
                                                             output &
  0 P2 2 P3 4 P1 5 P2 3 P1 9 Idle 10 P2 12 P3 14 Idle 15 P2 1
                                                             Enter number &
                                                              Enter eneutic
   Idle 20. P2.
                                    P2.
                          P2, P3
                                                              3 2 2
                                                              Enter deadline
                                                               20 5 10
                                     14 15
                                             14 20.
                           10
                                                               0 P2 2 P3
                                                               Idle 20 PL.
(1) void eden }
     int time = 1cm Array (all, n);
   Int op=0, pr=0, pre=-1;
    int glag, i;
      white (op <= time) of
        for (1=0; 1 < m ; 1++) d
           12 (0p 010 dl [i] ==0) $
               read (i)=1',
        Aag = 0;
       for (120; 12n; 4+) 1
          it ( read (i) = a) )
           flag=1;
        it (flag = =0) 1
           Pr = -1"
        yeese 1
           pr 2-1;
            for lise; len; i++)}
              i + (read (i) ==1) 1
                 1+ ( Ba = = -1 11 b(1) < b(ba) )1
                       De = = 1 %
```

