```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void sort (int pid[], int b[], int pt[], int n)
{
  int temp = 0;
  for (int i = 0; i < n; i++)
        {
          for (int j = i; j < n; j++)
                {
                  if (pt[j] < pt[i])
                        {
                          temp = pt[i];
                          pt[i] = pt[j];
                          pt[j] = temp;
                          temp = b[j];
                          b[j] = b[i];
                          b[i] = temp;
                          temp = pid[i];
                          pid[i] = pid[j];
                          pid[j] = temp;
                        }
                }
        }
}

int gcd (int a, int b)
{
  int r;
  while (b > 0)
        {
          r = a % b;
          a = b;
          b = r;
        }
  return a;
}

int lcm1 (int p[], int n)
{
  int lcm = p[0];
  for (int i = 1; i < n; i++)
        {
          lcm = (lcm * p[i]) / gcd (lcm, p[i]);
        }
  return lcm;
}


void main ()
```

```
{
  int n;
  printf ("Enter the number of processes:");
  scanf ("%d", &n);
  int pid[n], b[n], pt[n], rem[n];
  printf("Enter the PID,CPU burst time and time period");
  for(int i=0;i<n;i++)
  {
        scanf("%d%d%d",&pid[i],&b[i],&pt[i]);
        rem[i]=b[i]
  }
  sort (pid, b, pt, n);

  int l = lcm1 (pt, n);
  printf ("LCM=%d\n", l);

  printf ("\nRate Monotone Scheduling:\n");
  printf ("PID\t Burst\tPeriod\n");
  for (int i = 0; i < n; i++)
        printf ("%d\t\t%d\t\t%d\n", pid[i], b[i], pt[i]);


  double sum = 0.0;
  for (int i = 0; i < n; i++)
        {
          sum += (double) b[i] / pt[i];
        }
  double rhs = n * (pow (2.0, (1.0 / n)) - 1.0);
  printf ("\n%lf <= %lf =>%s\n", sum, rhs, (sum <= rhs) ? "true" : "false");
  if (sum > rhs)
        exit (0);

  printf ("Scheduling occurs for %d ms\n\n", l);


  int time = 0, prev = 0, x = 0;
  while (time < l)
        {
          int f = 0;
          for (int i = 0; i < n; i++)
                {
                  if (time % pt[i] == 0)
                        rem[i] = b[i];
                  if (rem[i] > 0)
                        {
                          if (prev != pid[i])
                                {
                                  printf ("%dms onwards: Process %d running\n",
time,
                                                pid[i]);
                                  prev = pid[i];
                                }
                        rem[i]--;
                        f = 1;
                        break;
                        x = 0;
                      }
              }
          if (!f)
              {
                if (x != 1)
                      {
                        printf ("%dms onwards: CPU is idle\n", time);
                        x = 1;
                      }
              }
          time++;
        }
}
```