OS - RATE MONOTONIC SCHEDULING

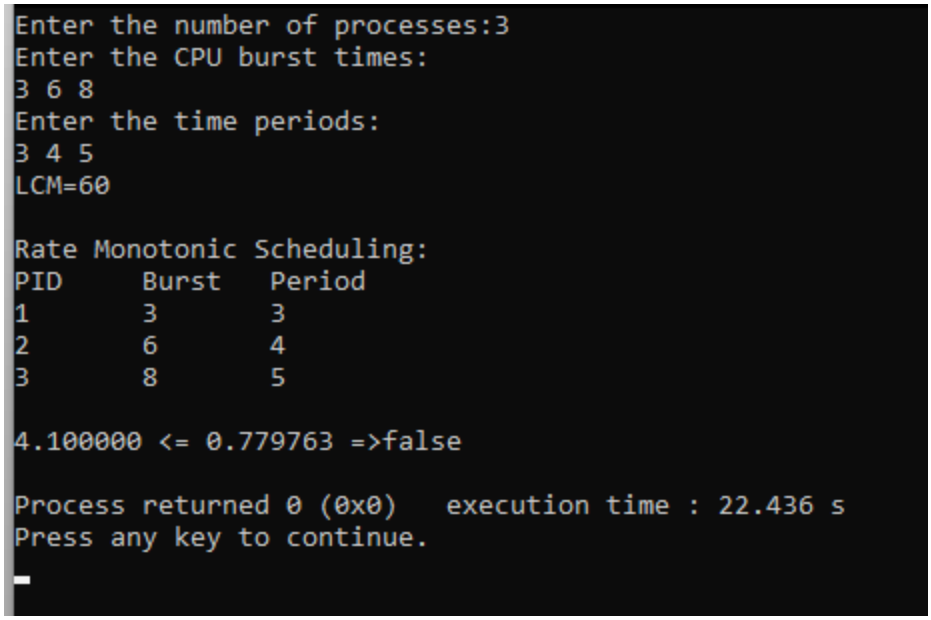
Write a C program to simulate Real-Time CPU Scheduling algorithms:
a) Rate- Monotonic

Code:

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

#define MAX 10

struct Process {
    int pid;
    int burst;
    int period;
};

int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int lcm(int a, int b) {
    return (a * b) / gcd(a, b);
}

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

int main() {
    int n;
    struct Process p[MAX];

    printf("Enter the number of processes:");
```

```
    scanf("%d", &n);

    printf("Enter the CPU burst times:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &p[i].burst);
        p[i].pid = i + 1;
    }

    printf("Enter the time periods:\n");
    int periods[MAX];
    for (int i = 0; i < n; i++) {
        scanf("%d", &p[i].period);
        periods[i] = p[i].period;
    }

    int lcm_val = find_lcm(periods, n);
    printf("LCM=%d\n\n", lcm_val);

    printf("Rate Monotonic Scheduling:\n");
    printf("PID\tBurst\tPeriod\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\n", p[i].pid, p[i].burst, p[i].period);
    }

    double utilization = 0;
    for (int i = 0; i < n; i++) {
        utilization += (double)p[i].burst / p[i].period;
    }

    double bound = n * (pow(2.0, 1.0 / n) - 1);

    printf("\n%.6f <= %.6f =>%s\n", utilization, bound, (utilization <= bound) ? "true" :
"false");

    return 0;
}
```

Output:


Enter the number of processes:3
Enter the CPU burst times:
3 6 8
Enter the time periods:
3 4 5
LCM=60
Rate Monotonic Scheduling:
PID Burst Period
1 3 3
2 6 4
3 8 5
4.100000 <= 0.779763 =>false
Process returned 0 (0x0) execution time : 22.436 s
Press any key to continue.