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

a) Rate- Monotonic

```
#include <stdio.h>

struct Process {
   int id;
   int burst_time;
   int period;
   int deadline;
   int remaining_time;
};

int main() {
   int n, time = 0, hyper_period = 1;
   struct Process p[10];

   printf("Enter number of real-time processes: ");
   scanf("%d", &n);

   for(int i = 0; i < n; i++) {
      printf("Enter burst time and period for P[%d]: ", i+1);
      scanf("%d %d", &p[i].burst_time, &p[i].period);
      p[i].id = i+1;
      p[i].remaining_time = p[i].burst_time;
      p[i].deadline = p[i].period;

      // Calculate LCM (simplified as product for hyper-period)
      hyper_period *= p[i].period;
   }

   printf("\nScheduling up to time %d (hyper-period):\n", hyper_period);
   printf("Time\tProcess\n");

   while(time < hyper_period) {
      int shortest = -1;
      for(int i = 0; i < n; i++) {
         if(time % p[i].period == 0 && p[i].remaining_time != 0) {
            p[i].remaining_time = p[i].burst_time;
            p[i].deadline = time + p[i].period;
         }
         if(p[i].remaining_time > 0) {
            if(shortest == -1 || p[i].period < p[shortest].period) {
               shortest = i;
            }
         }
```

```
        }

        if(shortest != -1) {
            printf("%d\tP[%d]\n", time, p[shortest].id);
            p[shortest].remaining_time--;
        } else {
            printf("%d\tIdle\n", time);
        }

        time++;
    }

    return 0;
}
```

Output

```
Enter number of real-time processes: 2
Enter burst time and period for P[1]: 2
3
Enter burst time and period for P[2]: 3
2

Scheduling up to time 6 (hyper-period):
Time    Process
0   P[2]
1   P[2]
2   P[2]
3   P[2]
4   P[2]
5   P[2]


=== Code Execution Successful ===
```