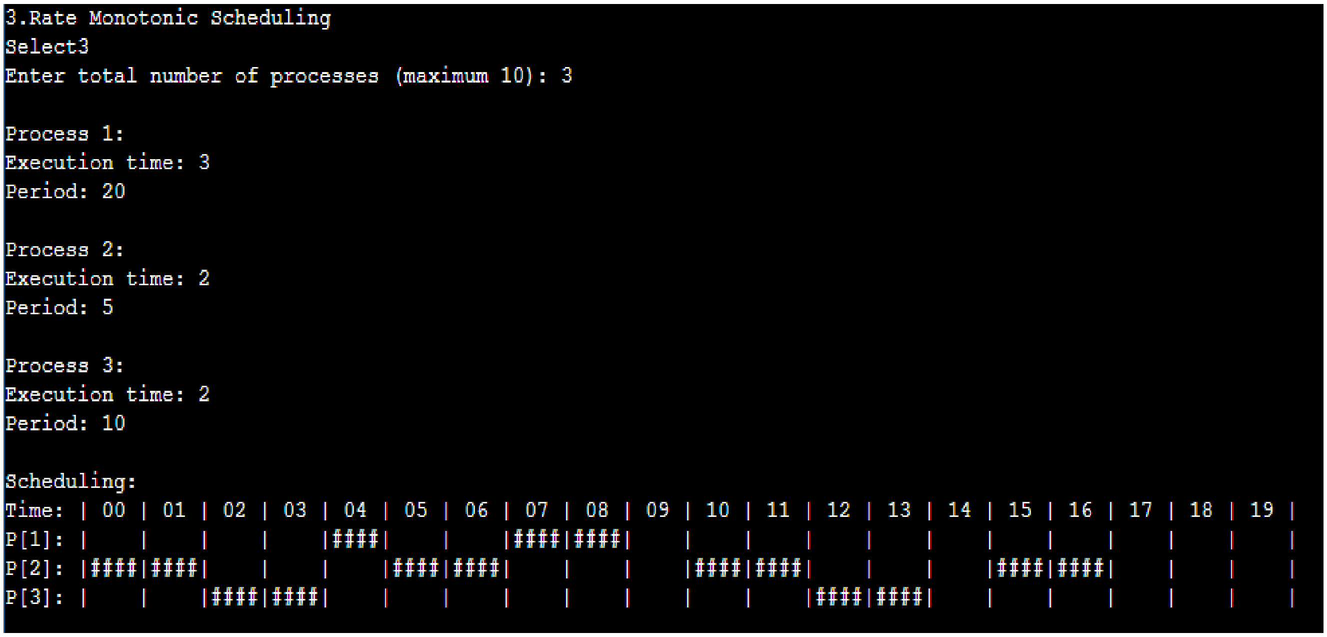
Rate-Monotonic Sheduling

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
#define MAX_PROCESS 10

int num_of_process = 3, count, remain, time_quantum;
int execution_time[MAX_PROCESS], period[MAX_PROCESS], remain_time[MAX_PROCESS],
deadline[MAX_PROCESS], remain_deadline[MAX_PROCESS];
int burst_time[MAX_PROCESS], wait_time[MAX_PROCESS],
completion_time[MAX_PROCESS], arrival_time[MAX_PROCESS];
void get_process_info(int selected_algo)
{
    printf("Enter total number of processes (maximum %d): ", MAX_PROCESS);
    scanf("%d", &num_of_process);
    if (selected_algo == 2)
    {
        printf("\nEnter Quantum time: ");
        scanf("%d", &time_quantum);
        if (time_quantum < 1)
        {
            printf("Invalid Input: Time quantum should be greater than 0\n");
            exit(0);
        }
    }
    for (int i = 0; i < num_of_process; i++)
    {
        printf("\nProcess %d:\n", i + 1);
        if (selected_algo == 1)
        {
            printf("Burst time: ");
            scanf("%d", &burst_time[i]);
        }
        else if (selected_algo == 2)
        {
            printf("Arrival Time: ");
            scanf("%d", &arrival_time[i]);
            printf("Burst Time: ");
            scanf("%d", &burst_time[i]);
            remain_time[i] = burst_time[i];
        }
        else if (selected_algo > 2)
        {
            printf("Execution time: ");
            scanf("%d", &execution_time[i]);
            remain_time[i] = execution_time[i];
            if (selected_algo == 4)
            {
                printf("Deadline: ");
                scanf("%d", &deadline[i]);
            }
            else
            {
                printf("Period: ");
                scanf("%d", &period[i]);
            }
        }
    }
}
int max(int a, int b, int c)
{
    int max;
    if (a >= b && a >= c)
        max = a;
    else if (b >= a && b >= c)
        max = b;
    else if (c >= a && c >= b)
        max = c;
    return max;
}
int get_observation_time(int selected_algo)
{
    if (selected_algo < 3)
    {
        int sum = 0;
        for (int i = 0; i < num_of_process; i++)
        {
            sum += burst_time[i];
        }
        return sum;
    }
    else if (selected_algo == 3)
    {
        return max(period[0], period[1], period[2]);
    }
    else if (selected_algo == 4)
    {
        return max(deadline[0], deadline[1], deadline[2]);
    }
}

void print_schedule(int process_list[], int cycles)
{
    printf("\nScheduling:\n");
    printf("Time: ");
    for (int i = 0; i < cycles; i++)
    {
        if (i < 10)
            printf("| 0%d ", i);
        else
            printf("| %d ", i);
    }
    printf("|\n");
    for (int i = 0; i < num_of_process; i++)
    {
        printf("P[%d]: ", i + 1);
        for (int j = 0; j < cycles; j++)
        {
            if (process_list[j] == i + 1)
                printf("|####");
            else
                printf("|    ");
        }
        printf("|\n");
    }
}

void rate_monotonic(int time)
{
    int process_list[100] = {0}, min = 999, next_process = 0;
    float utilization = 0;
    for (int i = 0; i < num_of_process; i++)
    {
        utilization += (1.0 * execution_time[i]) / period[i];
    }
    int n = num_of_process;
    if (utilization > n * (pow(2, 1.0 / n) - 1))
    {
        printf("\nGiven problem is not schedulable under the said scheduling algorithm.\n");
        exit(0);
    }
    for (int i = 0; i < time; i++)
    {
        min = 1000;
        for (int j = 0; j < num_of_process; j++)
        {
            if (remain_time[j] > 0)
            {
                if (min > period[j])
                {
                    min = period[j];
                    next_process = j;
                }
            }
        }
        if (remain_time[next_process] > 0)
        {
            process_list[i] = next_process + 1;
            remain_time[next_process] -= 1;
        }
        for (int k = 0; k < num_of_process; k++)
        {
            if ((i + 1) % period[k] == 0)
            {
                remain_time[k] = execution_time[k];
                next_process = k;
            }
        }
    }
    print_schedule(process_list, time);
}
int main(int argc, char *argv[])
{
    int option = 0;
    printf("3.Rate Monotonic Scheduling\n");
    printf("Select");
    scanf("%d", &option);
    get_process_info(option);
    int observation_time = get_observation_time(option);

     if (option == 3)
        rate_monotonic(observation_time);
    return 0;
}
```

OUTPUT :



```
3.Rate Monotonic Scheduling
Select3
Enter total number of processes (maximum 10): 3

Process 1:
Execution time: 3
Period: 20

Process 2:
Execution time: 2
Period: 5

Process 3:
Execution time: 2
Period: 10

Scheduling:
Time: | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
P[1]: |    |    |    |    |####|    |    |####|####|    |    |    |    |    |    |    |    |    |    |    |
P[2]: |####|####|    |    |    |####|####|    |    |    |####|####|    |    |    |####|####|    |    |    |
P[3]: |    |    |####|####|    |    |    |    |    |    |    |    |####|####|    |    |    |    |    |    |
```