RATE MONOTONIC SCHEDULING

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

#define MAX_PROCESS 10

int num_of_process = 3;
int execution_time[MAX_PROCESS], period[MAX_PROCESS],
remain_time[MAX_PROCESS];

// collecting details of processes
void get_process_info()
{
   printf("Enter total number of processes (maximum %d): ",
MAX_PROCESS);
   scanf("%d", &num_of_process);
   if (num_of_process < 1)
   {
      printf("Do you really want to schedule %d processes? -_-\n",
num_of_process);
      exit(0);
   }
   for (int i = 0; i < num_of_process; i++)
   {
      printf("\nProcess P%d: \n", i + 1);

      printf("> Execution time: ");
      scanf("%d", &execution_time[i]);
      remain_time[i] = execution_time[i];
```

```
        printf("> Period: ");
        scanf("%d", &period[i]);
    }
}

// get maximum of three numbers
int max(int a, int b, int c)
{
    if (a >= b && a >= c)
        return a;
    else if (b >= a && b >= c)
        return b;
    else
        return c;
}

// calculating the observation time for scheduling timeline
int get_observation_time()
{
    return max(period[0], period[1], period[2]);
}

// print scheduling sequence
void print_schedule(int process_list[], int cycles)
{
    printf("\nScheduling:-\n\n");
    printf("Time: ");
    for (int i = 0; i < cycles; i++)
    {
        if (i < 9)
            printf("| 0%d ", i + 1);
        else
            printf("| %d ", i + 1);
    }
    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)
{
    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 said scheduling
algorithm.\n");
        exit(0);
    }

    int process_list[time];
    int min = 999, next_process = 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; // +1 for catering 0 array index.
            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[])
{
    printf("\nRate Monotonic Scheduling\n");
    printf("------------------------\n");
```

```
    get_process_info(); // collecting processes detail
    int observation_time = get_observation_time();

    rate_monotonic(observation_time);

    return 0;
}
```

OUTPUT: