**3. Write a C program to simulate Real-Time CPU Scheduling algorithms: a) Rate- Monotonic  
b) Earliest-deadline First**

#include <stdio.h>

#include <math.h>

#define MAX 10

struct Task {

int id, burst, period, deadline;

int rem, next\_dl;

};

int lcm(int a, int b) {

int max = (a > b) ? a : b;

while (1) {

if (max % a == 0 && max % b == 0) return max;

++max;

}

}

int lcm\_all(int arr[], int n) {

int res = arr[0];

for (int i = 1; i < n; i++)

res = lcm(res, arr[i]);

return res;

}

void rate\_monotonic(struct Task t[], int n, int end) {

printf("\nPID\tBurst\tPeriod\n");

for (int i = 0; i < n; i++)

printf("%d\t%d\t%d\n", t[i].id, t[i].burst, t[i].period);

float u = 0;

for (int i = 0; i < n; i++)

u += (float)t[i].burst / t[i].period;

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

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

for (int time = 0; time < end; time++) {

for (int i = 0; i < n; i++)

if (time % t[i].period == 0)

t[i].rem = t[i].burst;

int run = -1;

for (int i = 0; i < n; i++)

if (t[i].rem > 0)

if (run == -1 || t[i].period < t[run].period)

run = i;

if (run != -1) t[run].rem--;

}

}

void edf(struct Task t[], int n, int end) {

printf("\nPID\tBurst\tDeadline\n");

for (int i = 0; i < n; i++)

printf("%d\t%d\t%d\n", t[i].id, t[i].burst, t[i].deadline);

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

for (int time = 0; time < end; time++) {

for (int i = 0; i < n; i++)

if (time % t[i].period == 0) {

t[i].rem = t[i].burst;

t[i].next\_dl = time + t[i].deadline;

}

int run = -1;

for (int i = 0; i < n; i++)

if (t[i].rem > 0)

if (run == -1 || t[i].next\_dl < t[run].next\_dl)

run = i;

if (run != -1) {

printf("%dms : Task %d is running.\n", time, t[run].id);

t[run].rem--;

} else {

printf("%dms : CPU is idle.\n", time);

}

}

}

int main() {

int n, periods[MAX];

struct Task t[MAX], rms[MAX], edf\_tasks[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", &t[i].burst);

printf("Enter the deadlines:\n");

for (int i = 0; i < n; i++)

scanf("%d", &t[i].deadline);

printf("Enter the time periods:\n");

for (int i = 0; i < n; i++) {

scanf("%d", &t[i].period);

t[i].id = i + 1;

periods[i] = t[i].period;

}

int end = lcm\_all(periods, n);

printf("LCM=%d\n", end);

for (int i = 0; i < n; i++) {

rms[i] = t[i]; rms[i].rem = 0;

edf\_tasks[i] = t[i]; edf\_tasks[i].rem = 0; edf\_tasks[i].next\_dl = 0;

}

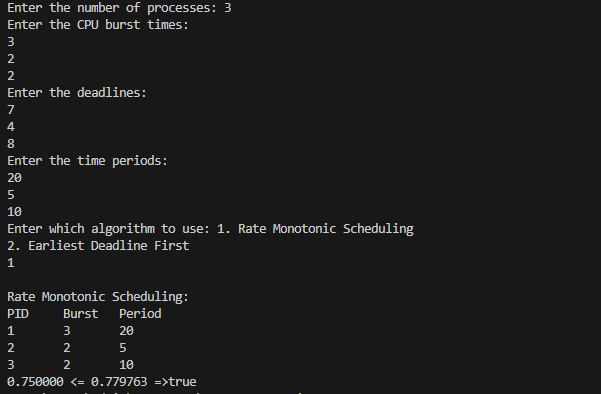
rate\_monotonic(rms, n, end);

edf(edf\_tasks, n, end);

return 0;

}

Rate Monotonic Output



Earliest DeadLine First Output

