PAGE NO :
DATE :

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

a) Rate-Monotonic

```
#include <stdio.h>
#define MAX_TASKS 100

typedef struct {
    int pid;
    int period;
    int exec_time;
    int deadline;
} Task;

float cpu_utilization (Task tasks[], int n) {
    float total_util = 0.0;
    for(int i=0; i<n; i++) {
    float task_util = (float) task[i].exectime / tasks[i].
                                                    period;
    total_util += task_util;
    }
    float cpu_util = total_util * 100;
        return cpu_util;
    }

Void rateMonotonic() {
    int n, i;
    printf("Enter the number of tasks:");
    scanf("%d", &n);
    Task tasks[MAX_TASKS];

for(i=0; i<n; i++) {
    printf("Task %d\n", i+1);
```

```
printf("Enter period: ");
scanf("%d", &tasks[i].period);
printf("Enter execution time: ");
scanf("%d", &tasks[i].exec_time);
printf("Enter deadline: ");
scanf("%d", &tasks[i].deadline);
tasks[i].pid = i+1;
}
float cpu_util = cpu_utilization(tasks, n);
printf("cpu utilization: %.4f %%\n", cpu_util
}

Void main(){
int choice, n, i;
printf("1. Rate monotonic\n2. exit\n\n");
while(1){
scanf("%d", &choice);
switch(choice){
case 1: rateMonotonic();
break;
case 2: exit(0);
default: printf("Wrong choice\n");
}
}
}
```

Output

1. Rate Monotonic
2. Exit.

1. Enter the number of tasks : 3

Task 1

Enter period : 20

Enter execution time : 3

Enter deadline : 20

Task 2

Enter period : 5

Enter execution time : 2

Enter deadline : 5

Task 3.

Enter period : 10

Enter execution time : 2

Enter deadline : 10

CPU utilization : 75.0000%

Gantt chart

| P2 | P3 | P1 | P2 | P1 | P1 | idle | P2 | P3 | idle | P2 | idle |

0 2 4 5 7 8 9 10 12 14 15 17 20

```
Enter the number of tasks: 3
Enter the execution time and period for task 1: 1 10
Enter the execution time and period for task 2: 2 5
Enter the execution time and period for task 3: 3 20

Task2 -> starts:0.000, ends:2.000, execution time:2.000, period:5.000
Task1 -> starts:2.000, ends:3.000, execution time:1.000, period:10.000
Task3 -> starts:3.000, ends:6.000, execution time:3.000, period:20.000
```

DATE : 19/7/23

Week-5

1) E-D.F

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

int et[10], i, n, dl[10], p[10], ready[10], flag=1;

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

Void edp() {
    int time = lcmArray(dl, n);
    int op = 0, pr=0, pre=-1;
    int flag, i;

    while (op <= time) {
        for(i=0;i<n;i++) {
            if (op % dl[i]==0) {
```

```
        ready[i] = 1;
    }
}

flag = 0;
for (i = 0; i < n; i++) {
    if (ready[i] == 1) {
        flag = 1;
        break;
    }
}

if (flag == 0) {
    pr = -1;
} else {
    pr = -1;
    for (i = 0; i < n; i++) {
        if (ready[i] == 1) {
            if (pr == -1 || p[i] < p[pr]) {
                pr = i;
            }
        }
    }
}

if (pr != pre) {
    if (pr == -1) {
        printf("%d Idle", op);
    } else {
        printf("%d P%d", op, pr+1);
    }
}

op++;
if (pr != -1)
    p[pr] = p[pr] - 1;
```

```
        if (p[pr] == 0) {
            p[pr] = et[pr];
            ready[pr] = 0;
        }
    }
    pre = pr;
}
printf("\n");
}

int main() {
    int ch, k = 1;
    while (k) {

printf("Enter your choice : \n1. Monotonic \n2. EDF \n3
        progorithmic \n4. Exit \n");
Scanf("%d", &ch);
if (ch == 4)
    exit(0);
printf("Enter the number of processes:");
Scanf("%d", &n);
printf("Enter execution time: \n");
    for (i=0; i<n; i++)
        Scanf("%d", &et[i]);
printf("Enter deadline: \n");
    for (i=0; i<n; i++)
        Scanf("%d", &dl[i]);
    for (i=0; i<n; i++)
        p[i] = et[i];
    for (i=0; i<n; i++)
        ready[i] = 0;
    switch (ch) {
    case 1: mono();
            break;
```

```
case 2:
        edf();
        break;
case 3:
        prop();
        break;
case 4:
        k=0;
        break;
default:
        printf("Invalid choice.\n"); } }

return 0;
}
```

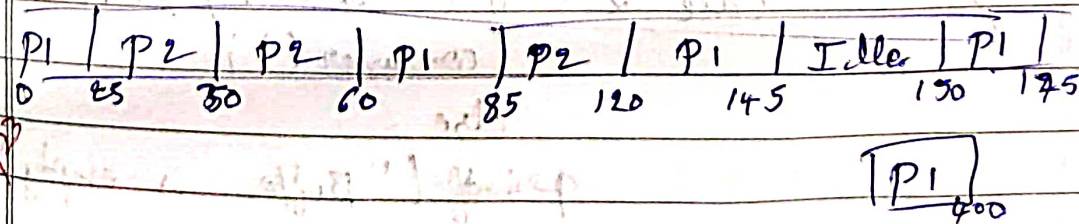
output

1. EDF
2. Exit

1.

Enter the number of processes : 2
Enter execution time : 25 35
Enter deadline : 50 80

0 P1 25 P2 60 P1 85 P2 120 P1 145 Idle 150 P1 175 P2 210 P1 235 Idle 240 P2 250 P1 275 P2 300 P1 325 P2 360 P1 385 Idle 400 P1



9/10

21/7/23

```
Enter number of tasks:3
Enter Task 1 parameters
Arrival time: 0
Execution time: 2
Deadline time: 5
Period: 5
Enter Task 2 parameters
Arrival time: 0
Execution time: 3
Deadline time: 8
Period: 8
Enter Task 3 parameters
Arrival time: 0
Execution time: 1
Deadline time: 10
Period: 10
CPU Utilization 0.875000
```

```
Tasks can be scheduled
0  Task 1
1  Task 1
2  Task 2
3  Task 2
4  Task 2
5  Task 1
6  Task 1
7  Task 3
8  Task 2
9  Task 2
10  Task 1
11  Task 1
12  Task 2
13  Task 3
14  Idle
15  Task 1
16  Task 1
17  Task 2
18  Task 2
19  Task 2
20  Task 1
21  Task 1
22  Task 3
23  Idle
24  Task 2
25  Task 1
26  Task 1
27  Task 2
28  Task 2
29  Idle
30  Task 1
31  Task 1
32  Task 2
33  Task 2
34  Task 2
35  Task 1
36  Task 1
37  Task 3
38  Idle
39  Idle
40  Task 1
```

```
Initial Numbers: 3 4 6 5 4
Decrementing number at index 4: 3 4 6 5 3
Decrementing number at index 0: 2 4 6 5 3
Decrementing number at index 4: 2 4 6 5 2
Decrementing number at index 2: 2 4 5 5 2
Decrementing number at index 1: 2 3 5 5 2
Decrementing number at index 0: 1 3 5 5 2
Decrementing number at index 2: 1 3 4 5 2
Decrementing number at index 1: 1 2 4 5 2
Decrementing number at index 1: 1 1 4 5 2
Decrementing number at index 2: 1 1 3 5 2
Decrementing number at index 4: 1 1 3 5 1
Decrementing number at index 3: 1 1 3 4 1
Decrementing number at index 2: 1 1 2 4 1
Decrementing number at index 2: 1 1 1 4 1
Decrementing number at index 3: 1 1 1 3 1
Decrementing number at index 0: 0 1 1 3 1
Decrementing number at index 2: 0 1 0 3 1
Decrementing number at index 1: 0 0 0 3 1
Decrementing number at index 4: 0 0 0 3 0
Decrementing number at index 3: 0 0 0 2 0
Decrementing number at index 3: 0 0 0 1 0
Decrementing number at index 3: 0 0 0 0 0
All numbers reached 0.
```