12/6/24

PAGE NO.:
DATE / /

1) Write a C program to stimulate Real Time CPU scheduling Algorithms.

a) Rate - Monotonic:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void sort (int proc[], int b[], int pt[], int n){
    int temp=0;
    for (int i=0 ;i<n ;i++){
        for (int j=0 ;j<n ;j++){
            if (pt[j]<pt[i]){
                temp = pt[i];
                pt[i] = pt[j];
                pt[j] = temp;
                temp = b[j];
                b[j] = b[i];
                b[i] = temp;
                temp = proc[i];
                proc[i] = proc[j];
                proc[j] = temp;
            }
        }
    }
}
int gcd (int a, int b){
    int r;
    while (b>0){
        r = a %b;
        a=b
        b=r;
    }
    return a;
}
```

```
int lcmul (int p[], int n){
    int lcm = p[0];
    for (int i=0; i<n; i++){
        lcm = (lcm * p[i]) / gcd (lcm, p[i]);
    }
    return lcm;
}
void main(){
    int n;
    printf (" Enter nor of processes: ");
    scanf ("%d", &n);
    int proc[n], b[n], pt[n], rem[n];
    printf (" Enter CPU burst time \n");
    for (int i=0; i<n; i++){
        scanf ("%d", &b[i]);
        rem[i] = b[i];
    }
    printf (" Enter the time periods \n");
    for (int i=0; i<n; i++)
        scanf ("%d", &pt[i]);
    for (int i=0; i<n; i++)
        proc[i] = i+1;
    sort (proc, b, pt, n);
    int l = lcmul (pt, n);
    printf (" LCM = %d \n", l);
    printf ("\n Rate Monotone Scheduling: \n");
    printf (" PID \t Burst \t Period \n");
    for (int i=0; i<n; i++)
        printf ("%d \t %d \t %d \t \n", proc[i], b[i], pt[i]);
    double sum = 0.0;
    for (int i=0; i<n; i++){
        sum += (double) b[i] / pt[i];
    }
```

```
double rhs = n * (pow (2.0, (1.0/n)) - 1.0);
printf ("\n %f <= %f = %s \n", sum, rhs, (sum <= rhs)
        ? "true" : "false");
if (sum > rhs)
    exit(0);
printf ("Scheduling occurs %d ms \n", l);
int time = 0, prev=0, x=0;
while (time < l) {
    int f = 0;
    for (int i = 0; i < n; i++) {
        if (time % pt[i] == 0)
            rem[i] = b[i];
        if (rem[i] > 0) {
            if (prev != proc[i]) {
                printf ("%d ms onwards: Process %d running \n",
                        time, proc[i]);
                prev = proc[i];
            }
            rem[i]--;
            f = 1;
            break;
            x = 0;
        }
    }
    if (!f) {
        if (x != 1) {
            printf ("%d ms onwards: CPU is idle \n",
            time);
            x = 1;
        }
    }
    time++;
}
```

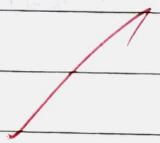
PAGE NO.:
DATE / /

→ Output:

Enter the nor of process : 3
Enter the CPU burst time: 3 2 2
Enter the time period: 20 5 10
LCM = 20
Rate Monotone Scheduling:

PID	Burst	Period
2	2	5
3	2	10
1	3	20

0.75000 <= 0.779763 ⇒ true
Scheduling occurs for 20 ms
0ms onwards: Process 2 running
2 ms onwards: Process 3 running
4ms onwards: Process 1 running
5ms onwards: Process 2 running
7ms onwards: Process 1 running
8ms onwards: CPU is idle
10ms onwards: Process 2 running



6) Earliest - Deadline First;

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void sort (int proc[], int d[], int b[], int pt[], int n){
    int temp=0;
    for (int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            if (d[j]<d[i]){
                temp = d[j];
                d[j] = d[i];
                d[i] = temp;
                temp = pt[i];
                pt[i] = pt[j];
                pt[j] = temp;
                temp = ~~proc~~ b[j];
                b[j] = b[i];
                b[i] = temp;
                temp = proc[i];
                proc[i] = proc[j];
                proc[j] = ~~proc[i]~~; temp;
            }
        }
    }
}
int gcd (int a, int b){
    int r;
    while (b>0){
        r = a%b;
        a = b;
        b = r;
    } return a;
}
```

```
int lcmul (int p[], int n){
    int lcm = p[0];
    for (int i = 1; i<n ; i++){
        lcm = (lcm * p[i]) /gcd (lcm, p[i]);
    }
    return lcm;
}
void main(){
    int n;
    printf ("Enter the nor of process :");
    scanf ("%d", &n);
    int proc[n], b[n], pt[n], d[n], rem[n];
    printf ("Enter the CPU burst time:\n");
    for(int i=0 ; i<n; i++){
        scanf ("%d", &b[i]);
        rem[i] = b[i];
    }
    printf ("Enter deadlines");
    for (i=0; i<n; i++){
        scanf ("%d", &d[i]);
        scanf ("%d", &pt[i]);
        proc[i] = i+1;
    }
    sort (proc, d, b, pt, n);
    int l = lcmul (pt, n);
    printf ("Earliest Deadline Scheduling:\n");
    printf ("PID \t Burst \t Deadline \t period \n");
    for (i=0; i<n; i++)
        printf ("%d \t %d \t %d \t %d \t \n", proc[i],
                b[i], d[i], pt[i]);
    printf ("Scheduling occurs for %d ms \n", l);
    int time = 0, prev = 0, x = 0;
```

```
int ndl[n];
for (int i=0; i<n; i++) {
        ndl[i] = d[i];
        rem[i] = b[i];
}
while (time<l) {
    for (int i=0; i<n; i++){
        if (time % pt[i] == 0 && time != 0) {
            ndl[i] = time + d[i];
            rem[i] = b[i];
        }
    }
    int mdl = l+1;
    int toE = -1;
    for (int i=0; i<n; i++){
        if (rem[i] > 0 && ndl[i] < mdl) {
            mdl = ndl[i];
            toE = i;
        }
    }
    if (toE != -1) {
        printf("%d ms : Task %d is running.\n", time,
        proc[toE]);
        rem[toE]--;
    }
    else {
        printf("%d ms : CPU is idle.\n", time);
    }
    time++;
}
}
```

PAGE NO.:
DATE / /

→ Output:

Enter no of processes: 3
Enter CPU burst time: 3 2 2
Enter the deadline: 7 4 8
Enter the time period: 20 5 10
Earliest Deadline Scheduling:

PID	Burst	Deadline	Period
2	2	4	5
1	3	7	20
3	2	8	10

Scheduling occurs for 20 ms
0ms: Task 2 is running
1ms: Task 2 is running
2ms: Task 1 is running
3ms: Task 1 is running
4ms: Task 1 is running
5ms: Task 3 is running
6ms: Task 3 is running
7ms: Task 2 is running
8ms: Task 2 is running
9ms: CPU is idle
10ms: Task 2 is running
11ms: Task 2 is running
12ms: Task 3 is running
13ms: Task 3 is running
14ms: CPU is idle
15ms: Task 2 is running
16ms: Task 2 is running
17ms: CPU is idle
18ms: CPU is idle
19ms: CPU is idle

PAGE NO.:
DATE / /

c) proportional Scheduling:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main (){
    srand (time (NULL));
    int n;
    printf (" Enter nor of processes: \n");
    scanf ("%d", &n);
    int p[n], t[n], cum[n], m[n];
    int c=0;
    int total=0, count=0;
    printf (" Enter tickets of processes: \n");
    for (int i=0; i<n; i++){
        scanf ("%d", &t[i]);
        c+= t[i];
        cum[i] = c;
        p[i] = i+1;
        m[i] = 0;
        total += t[i];
    }
    while (count <n){
        int wt = rand() % total;
        for (int i=0; i<n; i++){
            if (wt < cum[i] && m[i]==0){
                printf (" The winning nor is %d & winning
                    participant is: %d \n", wt, p[i]);
                m[i] = 1;
                count++;
            }
        }
    }
    printf (" Probabilities: \n");
```

```
    for (int i=0; i<n; i++){
        printf("Prob of winning: %.2f %%\n", p[i], ((double)t[i]/total)*100);
    }
}
```

→ Output:

```
Enter nor of processes: 3
Enter tickets for processes: 10 20 30
The winning nor is 47 & winning participant is: 3
The winning nor is 22 & winning participant is: 2
The winning nor is 4 & winning participant is: 1

Probabilities:
The probability of p1 winning: 16.67
The probability of p2 winning: 33.33
The probability of p3 winning: 50.00
```

PAGE NO.:
DATE / /

2. Write a C program to stimulate producer-consumer problem using semaphores

```
#include <stdio.h>
void main(){
    int buffer[10], bufsize, in, out, produce, consume,
    choice=0;
    in=0, out=0, bufsize=10;
    while(choice!=3){
        printf("\n1. Produce \t 2. consume \t 3. Exit");
        printf("Enter your choice\n");
        scanf("%d", &choice);
        switch(choice){
            case 1: if((in+1)%bufsize==out)
                        printf("Buffer is full\n");
                    else{
                        printf("Enter value:\n");
                        scanf("%d", &produce);
                        buffer[in]=produce;
                        in=(in+1)%bufsize;
                    }
                    break;
            case 2: if(in==out)
                        printf("\n Buffer is Empty");
                    else{
                        consume=buffer[out];
                        printf("The consumed value is %d",
                            consume);
                        out=(out+1)%bufsize;
                    }
                    break;
        }
    }
}
```

→ Output :

1. Produce 2. Consume 3. Exit
Enter your choice : 1
Enter value : 1
Enter your choice : 1
Enter value : 2
Enter your choice : 2
Consumed value is 1
Enter your choice : 2
Consumed value is 2
Enter your choice : 2
Buffer is empty
Enter your choice : 3