**WEEK 4**

**OSLAB**

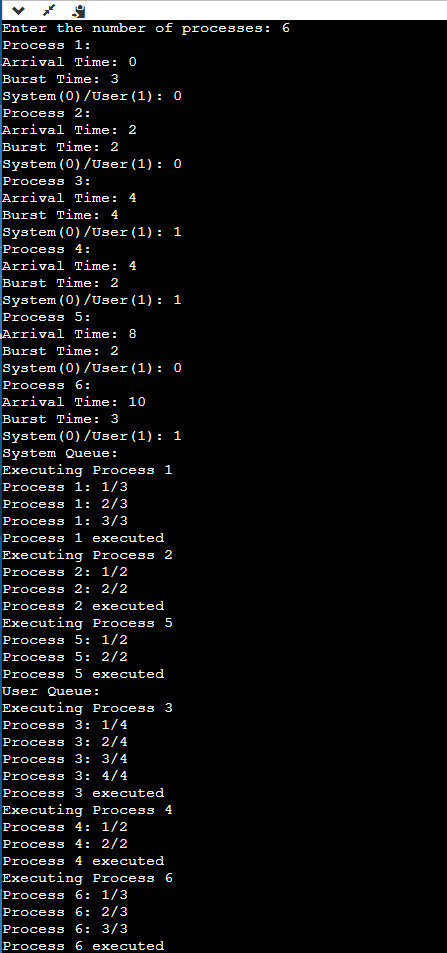
**1BM21CS255**

**CODE**

**PROG 1**

#include <stdio.h>  
  
#define MAX\_QUEUE\_SIZE 100  
  
// Structure to represent a process  
typedef struct {  
    int processID;  
    int arrivalTime;  
    int burstTime;  
    int priority; // 0 for system process, 1 for user process  
} Process;  
  
// Function to execute a process  
void executeProcess(Process process) {  
    printf("Executing Process %d\n", process.processID);  
    // Simulating the execution time of the process  
    for (int i = 1; i <= process.burstTime; i++) {  
        printf("Process %d: %d/%d\n", process.processID, i, process.burstTime);  
    }  
    printf("Process %d executed\n", process.processID);  
}  
  
// Function to perform FCFS scheduling for a queue of processes  
void scheduleFCFS(Process queue[], int size) {  
    for (int i = 0; i < size; i++) {  
        executeProcess(queue[i]);  
    }  
}  
  
int main() {  
    int numProcesses;  
    Process processes[MAX\_QUEUE\_SIZE];  
  
    // Reading the number of processes  
    printf("Enter the number of processes: ");  
    scanf("%d", &numProcesses);  
  
    // Reading process details  
    for (int i = 0; i < numProcesses; i++) {  
        printf("Process %d:\n", i + 1);  
        printf("Arrival Time: ");  
        scanf("%d", &processes[i].arrivalTime);  
        printf("Burst Time: ");  
        scanf("%d", &processes[i].burstTime);  
        printf("System(0)/User(1): ");  
        scanf("%d", &processes[i].priority);  
        processes[i].processID = i + 1;  
    }  
  
    // Separate system and user processes into different queues  
    Process systemQueue[MAX\_QUEUE\_SIZE];  
    int systemQueueSize = 0;  
    Process userQueue[MAX\_QUEUE\_SIZE];  
    int userQueueSize = 0;  
  
    for (int i = 0; i < numProcesses; i++) {  
        if (processes[i].priority == 0) {  
            systemQueue[systemQueueSize++] = processes[i];  
        } else {  
            userQueue[userQueueSize++] = processes[i];  
        }  
    }  
  
    // Execute system queue processes first  
    printf("System Queue:\n");  
    scheduleFCFS(systemQueue, systemQueueSize);  
  
    // Execute user queue processes  
    printf("User Queue:\n");  
    scheduleFCFS(userQueue, userQueueSize);  
  
    return 0;  
}

**OUTPUT**



**PROG 2**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

int n;

float e[20],p[20];

int i;

float ut,u,x,y;

printf("\n Enter Number of Processes :: ");

scanf("%d",&n);

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

{

printf("\n Enter Execution Time for P%d ::",(i+1));

scanf("%f",&e[i]);

printf("\n Enter Period for P%d ::",(i+1));

scanf("%f",&p[i]);

}

//calculate the utilization

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

{

x=e[i]/p[i];

ut+=x;

}

//calculate value of U

y=(float)n;

y=y\*((pow(2.0,1/y))-1);

u=y;

if(ut<u)

{

printf("\n As %f < %f ,",ut,u);

printf("\n The System is surely Schedulable");

}

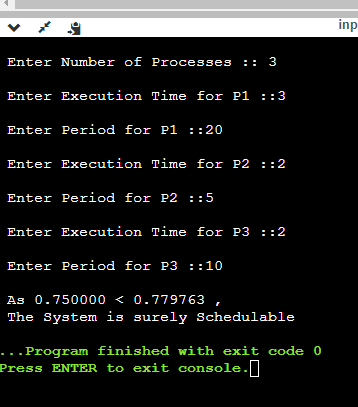
else

printf("\n Not Sure.....");

getch();

}

**OUTPUT**



**PROG 3**

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int n;

int period[MAX], execution[MAX], deadline[MAX];

int ready[MAX], task[MAX];

int time = 0;

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int gcd(int a, int b)

{

if (b == 0)

return a;

return gcd(b, a % b);

}

void sort() {

for (int i = 0; i < n - 1; i++) {

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

if (deadline[i] > deadline[j]) {

swap(&period[i], &period[j]);

swap(&execution[i], &execution[j]);

swap(&deadline[i], &deadline[j]);

}

}

}

}

int lcm(int arr[], int n)

{

int ans = arr[0];

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

ans = (((arr[i] \* ans)) / (gcd(arr[i], ans)));

return ans;

}

void schedule() {

int i, j;

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

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

ready[i] = 1;

}

}

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

if (ready[i] == 1) {

int min\_deadline = 1000000000;

int min\_index = -1;

for (j = 0; j < n; j++) {

if (ready[j] == 1 && deadline[j] < min\_deadline) {

min\_deadline = deadline[j];

min\_index = j;

}

}

task[min\_index] += execution[min\_index];

deadline[min\_index] += period[min\_index];

ready[min\_index] = 0;

}

}

}

int main() {

int total\_time;

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter the period, execution time and deadline of each process:\n");

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

scanf("%d %d %d", &period[i], &execution[i], &deadline[i]);

ready[i] = task[i] = 0;

}

sort();

printf("\nOrder of execution of processes in CPU timeline:\n");

total\_time = lcm(period, n);

while (time < total\_time) { // assuming total time is 100

schedule();

printf("%d ", task[0]);

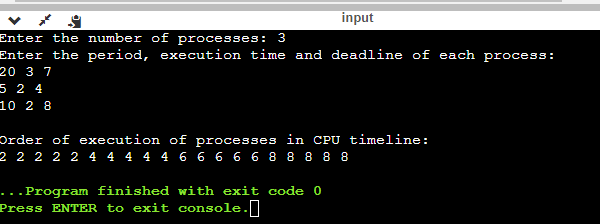
time++;

}

return 0;

}

**OUTPUT**

****