QUESTION NO 9:

DESCRIPTION:

Design a scheduler with multilevel queue having two queues which will schedule the processes on the basis of pre-emptive shortest remaining processing time first algorithm (SROT) followed by a scheduling in which each process will get 2 units of time to execute. Also note that queue 1 has higher priority than queue 2. Consider the following set of processes (for reference)with their arrival times and the CPU burst times in milliseconds.

Process	Arrival-Time	Burst-Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

Calculate the average turnaround time and average waiting time for each process. The input for number of processes and their arrival time, burst time should be given by the user.

ALGORITHM:

1. Take two queue's ready and ready 1.

2. In shortest job first scheduling algorithm the process with small Amount of time running until completion is selected to execute.
3.Enqueue processes p1 and p2 inside the ready queue according to the burst time of processes.
4.After completion of p1 and p2 processes we will enqueue our p3 And p4 processes in ready queue 1.
5. In RoundRobin Scheduling algorithm each process will assign a fixed time slot in cpu.
6. If the process completed its time quantum the then again sort the queue based on the priority of the processes allocated.
7. If the process completed execution before its time quantum then dequeue that process from the ready queue.
8. This process will continue until all the processes completed their execution.

EntireCode:

```
#include<stdio.h>

#include<conio.h>

int n=4;

int process[10][7];

int readyQueue[20],readyQueue1[20],burst[10],arrivalTime[10];

int front=0,rear=-1,max=-1,cpuTime=0,flag=0,from;


/*

1 arrival
```

```
2 burst
3 completion


*/
void Enqueue(int a){
	rear=rear+1;
	readyQueue[rear]=a;
}
int Dequeue(){
	int a=readyQueue[front];
	process[a-1][3]=cpuTime+1;
	front=front+1;
	flag=1;
	from=a;
	return a;
}

int IsAllExecuted(){
```

```
	// checking whether all the process has completed their
execution or not.....


	int i;
	for(i=0;i<2;i++){
		if(process[i][2]>0)
			break;
	}


	if(i<2)
	return 1;
	return 0;
}



int sortReadyQueue()
	{
		if(readyQueue[front]<=0)
		return -1;
```

```
            else if(readyQueue[front+1]<=0)
            return readyQueue[front]-1;
            else{
                int a=readyQueue[front]-1;
                int b=readyQueue[front+1]-1;
                if(process[a][2]<process[b][2])
                return a;
                readyQueue[front]=b+1;
                readyQueue[rear]=a+1;
                return readyQueue[front]-1;
            }
        }

void roundRobin(){
        int count=0,check=0;
        int cpuIdle=1;
        int i,id;
        int r=-1,f=0;
                for(i=2;i<n;i++){
```

```
                    r=r+1;
                    readyQueue1[r]=i+1;
                }
                if(cpuIdle==1)
                    id=readyQueue1[f]-1;
while(check==0){

//      if(cpuTime<=4){
            //}
        if(count==2){
            int a=readyQueue1[f];
            f=f+1;
            r=r+1;
            readyQueue1[r]=a;
            id=readyQueue1[f]-1;
            if(id!=-1){
                cpuIdle=0;
            }
            else
```

```
                    check=1;
                count=0;


            }
            cpuTime++;
            process[id][2]=process[id][2]-1;
            count++;
            if(process[id][2]<=0){
                f=f+1;
                process[id][3]=cpuTime+1;
                id=readyQueue1[f]-1;
                count=0;
                if(id==-1)
                check=1;
            }
        }
}
```

```
void scheduler(){
	int id,i,count=0;
	while(IsAllExecuted()){
		if(cpuTime<2)
		for(i=0;i<n;i++)
			if(cpuTime==arrivalTime[i])
				Enqueue(i+1);
		id=sortReadyQueue();
		if(id!=-1){
			process[id][2]=process[id][2]-1;
			count++;
			if(process[id][2]<=0){
				Dequeue();
			}
		}
		else{
			break;
		}
```

```
            cpuTime++;
        }

        roundRobin();
}

void calculateWaitingTime(){
        int i,avg=0;
        printf("\n==========================================
==========================================================
====================\n\n");
        printf("Process         WaitingTime\n");
        printf("-------------------------------\n");
        for(i=0;i<n;i++){
            process[i][5]=process[i][4]-burst[i];
            avg+=process[i][5];
            printf("  P%d                    %d\n",i+1,process[i][5]);
        }
        avg/=n;
        printf("\nThe Average Waiting Time is %d\n",avg);
```

```
}



void calculateTurnAroundTime(){
	int i,avg=0;
	printf("\n==========================================
==========================================
====================\n\n");
	printf("Process		TurnAroundTime\n");
	printf("----------------------------------\n");
	for(i=0;i<n;i++){
		process[i][4]=process[i][3]-process[i][1];
		avg+=process[i][4];
		printf("  P%d			%d\n",i+1,process[i][4]);
	}
	avg/=n;
	printf("\nThe Average TurnAround Time is %d\n",avg);


}
```

```
int main(){
	int i;
	for(i=0;i<n;i++){
		printf("Enter ArrivalTime of P%d :: ",i+1);
		scanf("%d",&arrivalTime[i]);
		printf("Enter BurstTime of P%d :: ",i+1);
		scanf("%d",&burst[i]);
		process[i][1]=arrivalTime[i];
		process[i][2]=burst[i];
	}
	printf("\n\n");
	scheduler();
	printf("\n");
	calculateTurnAroundTime();
	calculateWaitingTime();
	return 0;
}
```

CompleteSolution:-

1. **Completion Time:** Time at which process completes its execution.
2. **Turn Around Time:** Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. **Waiting Time(W.T):** Time Difference between turn around time and burst time. Waiting Time = Turn Around Time – Burst Time

QUESTION:

Process	Arrival-Time	Burst-Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

ArrivalTime of P1 :: 0

BurstTime of P1 :: 5

ArrivalTime of P2 :: 1

BurstTime of P2 :: 3

ArrivalTime of P3 :: 2

BurstTime of P3 :: 3

ArrivalTime of P4 :: 4

BurstTime of P4 :: 1

Process	TurnAroundTime
P1	8
P2	3
P3	11
P4	8

The Average TurnAround Time is 7.

Process	WaitingTime
P1	3
P2	0
P3	8
P4	7

The Average Waiting Time is 4.