Operating Systems:

Experiment 4

Name: Ayush Jain

SAP ID: 60004200132

Batch: B2

Computer Engineering

Aim: CPU scheduling algorithms like FCFS, SJF, Round Robin etc.

Problem Statement:

1) Perform comparative assessment of various Scheduling Policies like FCFS, SJF (preemptive and non-preemptive), Priority (preemptive and non-preemptive) and Round Robin.
2) Take the input processes, their arrival time, burst time, priority, quantum from user.

Description: Scheduling algorithms are used when more than one process is executable and the OS has to decide which one to run first.

Terms Used:

1) **Submit Time:** The process at which the process is given to CPU
2) **Burst Time:** The amount of time each process takes for execution
3) **Response Time:** The difference between the time when the process starts execution and the submit time.
4) **Turnaround Time:** The difference between the time when the process completes execution and the submit time.

First Come First Serve (FCFS)

The processes are executed in the order in which they have been submitted.

Shortest Job First (SJF)

The processes are checked at each arrival time and the process which have the shortest remaining burst time at that moment gets executed first. This is non-preemptive algorithm.

Priority Scheduling

Each process is assigned a priority and executable process with highest priority is allowed to run

Round Robin

Each process is assigned a time interval called its quantum (time slice)
If the process is still running at the end of the quantum the CPU is preempted and given to another process, and this continues in circular fashion, till all the processes are completely executed.

Code and Output:

1) **First Come First Serve (FCFS)**

```
import java.util.Scanner;
class FCFS
{
	public static void main(String args[])
	{
		Scanner sc = new Scanner(System.in);
		int n,i,j,temp,t1=0,t2=0,tat,wt;
		float atat=0,awt=0;
		System.out.print("Enter number of processes:
");		n=sc.nextInt();			int a[] = new int[n];
	int b[] = new int[n];		int c[] = new int[n];
int t[] = new int[n];		int w[] = new int[n];
		for(i=0; i<n; i++)
	{
			System.out.print("Enter arrival time of process "+(i+1)+": ");
			a[i]=sc.nextInt();
			System.out.print("Enter burst time of process "+(i+1)+": ");
			b[i]=sc.nextInt();
		}
		for(i=0; i<n-1; i++)
	{
			for(j=0; j<n-i-1; j++)
		{
				if(a[j+1]<a[j])
			{
					temp=a[j];
			a[j]=a[j+1];
	a[j+1]=temp;
temp=b[j];				b[j]=b[j+1];
```

```
b[j+1]=temp;                                }
}       }
                System.out.println("Process\t\tStart Time\tFinish Time");
                t1=a[0]; for(i=0;
i<n; i++)               {
                        t2=b[i]+t2;
                        System.out.println("  P"+(i+1)+"\t\t   "+t1+"\t\t   "+t2);
                        t1=t1+b[i];
        c[i]=t2;
                        t[i]=t2-a[i];
                        w[i]=t[i]-b[i];
        }
                System.out.println("Process\tA.T\tB.T\tC.T\tT.A.T\tW.T");
                for(i=0; i<n; i++)
                        System.out.println("
P"+(i+1)+"\t"+a[i]+"\t"+b[i]+"\t"+c[i]+"\t"+t[i]+"\t"+w[i]);
                for(i=0; i<n; i++)
                {
                        atat=atat+t[i];
                        awt=awt+w[i];
        }
                atat=atat/n;
                awt=awt/n;
                System.out.println("Average Turnaround Time: "+atat);
                System.out.println("Average Waiting Time: "+awt);
        }
}
```

```
Enter number of processes: 5
Enter arrival time of process 1: 0
Enter burst time of process 1: 4
Enter arrival time of process 2: 1
Enter burst time of process 2: 3
Enter arrival time of process 3: 2
Enter burst time of process 3: 1
Enter arrival time of process 4: 3
Enter burst time of process 4: 2
Enter arrival time of process 5: 4
Enter burst time of process 5: 5
Process         Start Time      Finish Time
  P1              0               4
  P2              4               7
  P3              7               8
  P4              8               10
  P5              10              15
Process A.T     B.T     C.T     T.A.T   W.T
  P1    0       4       4       4       0
  P2    1       3       7       6       3
  P3    2       1       8       6       5
  P4    3       2       10      7       5
  P5    4       5       15      11      6
Average Turnaround Time: 6.8
Average Waiting Time: 3.8
```

3) **Shortest Job First (SJF)**

```
import java.util.Scanner;
class ShortJobFirst
{
	public static void main(String args[])
	{
		Scanner sc = new Scanner(System.in);
		int n;
		System.out.print("Enter number of processes: ");
		n = sc.nextInt();
		int p[] = new int[n];    //Process id array		int
at[] = new int[n]; //Arrival time array		int bt[] = new
int[n]; //Burst time array		int flag[] = new int[n]; //To check
if process is completed
		int i, j, min1, min2, min3;
		for(i = 0; i < n; i++)
	{
			p[i] = i+1;
			System.out.print("Enter arrival time: ");
			at[i] = sc.nextInt();
  System.out.print("Enter burst time: ");    bt[i] = sc.nextInt();
			flag[i] = 0;
	}
		for(i = 0; i < n; i++)
	{
			min1 = at[i];
		min2 = bt[i];
	min3 = p[i];
j = i-1;
			while(j >= 0 && at[j] > min1)
	{
				at[j+1] = at[j];
			bt[j+1] = bt[j];
		p[j+1] = p[j];
				j--;
			}
			at[j+1] = min1;
		bt[j+1] = min2;
			p[j+1] = min3;
	}
		 int cur_t = 0; //Current time
```

```
                int st[] = new int[n];    //Starting time of each process int
                ct[] = new int[n]; //completion time array
                int tot = 0;       //To count number of processes completed int
                minbt = 1000; //To store the shortest bt and set to highest
value so that all bt values are compared
                int c = 0;         //To track id of process to be scheduled
next            while(tot < n)             {
                        for(i = 0; i < n; i++)
                {
                                if((at[i] <= cur_t) && (flag[i] == 0) && (bt[i] <= minbt))
                                {
                                        minbt = bt[i];
                                        c = i;
                        }
                        }
                        ct[c] = cur_t + minbt;
                st[c] = cur_t;
flag[c] = 1;                    cur_t = ct[c];
                        tot++;
  minbt = 1000; //reset so that bt values of remaining processes are compared   }
                int  tat[] = new int[n]; //Turnaround Time array
        int wt[] = new int[n]; //Waiting Time array
                float sum1 = 0, sum2 = 0; //sum1 for tat and sum2 for wt
System.out.println("\nProcess No.\tA.T\tB.T.\tC.T.\tT.A.T.\tW.T.");
for(i = 0; i < n; i++)                {
                        tat[i] = ct[i] - at[i];
        wt[i] = tat[i] - bt[i];
                        System.out.println("Process
"+p[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+tat[i]+"\t"+wt[i]);
                        sum1 += tat[i];
                        sum2 += wt[i];
        }
                System.out.println("Average Turn Around Time: "+(sum1/n));
                System.out.println("Average Waiting Time: "+(sum2/n));
                //To prepare Gantt Chart processes are soted according to start
time            for(i = 0; i < n; i++)                {
                        min1 = st[i];
                        min2 = ct[i];
                        min3 = p[i];
                j = i-1;
                        while(j >= 0 && st[j] > min1)
                        {
```

```
                    st[j+1] = st[j];
                ct[j+1] = ct[j];
                    p[j+1] = p[j];
                    j--;
                }
                st[j+1] = min1;
            ct[j+1] = min2;
                p[j+1] = min3;
        }
 System.out.println("\n\t\tGANTT CHART\nPROCESS \tStart Time\tCompletion Time");
            for(i = 0; i < n; i++)
        {
                System.out.println("PROCESS "+p[i]+"\t\t"+st[i]+"\t\t"+ct[i]);

            }
}
}
```

```
Enter number of processes: 5
Enter arrival time: 0
Enter burst time: 7
Enter arrival time: 2
Enter burst time: 5
Enter arrival time: 3
Enter burst time: 1
Enter arrival time: 4
Enter burst time: 2
Enter arrival time: 5
Enter burst time: 8

Process No.     A.T     B.T.    C.T.    T.A.T.  W.T.
Process 1       0       7       7       7       0
Process 2       2       5       15      13      8
Process 3       3       1       8       5       4
Process 4       4       2       10      6       4
Process 5       5       8       23      18      10
Average Turn Around Time: 9.8
Average Waiting Time: 5.2

                GANTT CHART
PROCESS         Start Time      Completion Time
PROCESS 1               0               7
PROCESS 3               7               8
PROCESS 4               8               10
PROCESS 2               10              15
PROCESS 5               15              23
```

3) Priority Scheduling

```
import java.util.Scanner;
    class Priority
    { public static void main(String args[]) {
                    Scanner sc = new Scanner(System.in); int
                    n;
                    System.out.print("Enter number of processes: "); n
                    = sc.nextInt();
                    int p[] = new int[n];                    //Process id array int
                    at[] = new int[n];              //Arrival time array int bt[]
                    = new int[n];           //Burst time array
                    int flag[] = new int[n];        //To check if process is
    completed           int priority[] = new int[n];            int i, j, min1, min2,
    min3, min4;
                    int cur_t = 0;                              //Current time
                    int st[] = new int[n];                  //Starting time of each process
                    int ct[] = new int[n];                  //completion time array
                    int tot = 0;                                //To count number of
    processes completed
                    int minpri = 1000;                           //To store the shortest bt
                    int c = 0;
                    int b[] = new int[n];
                    for(i = 0; i < n; i++)
            {
                            p[i] = i+1;
                            System.out.print("Enter arrival time: ");
                            at[i] = sc.nextInt();
                            System.out.print("Enter burst time: ");
                            bt[i] = sc.nextInt();
                            System.out.print("Enter priority [smallest number is high]: ");
                            priority[i] = sc.nextInt();
                            flag[i] = 0;
                            st[i] = -1;
                    }
                    for(i = 0; i < n; i++)
            {
                            min1 = at[i];
            min2 = bt[i];
    min3 = p[i];
                            min4 = priority[i];
                            j = i-1;
                            while(j >= 0 && at[j] > min1)
```

```
                {
                        at[j+1] = at[j];
                bt[j+1] = bt[j];
        p[j+1] = p[j];
                        priority[j+1] = priority[j];
                        j--;
                }
                at[j+1] = min1;
                bt[j+1] = min2;
                p[j+1] = min3;
                priority[j+1] = min4;
        }
        for(i = 0; i < n; i++)
                b[i] = bt[i];
        while(tot < n)
{
                for(i = 0; i < n; i++)
        {
                        if((priority[i] <= minpri) && (at[i] <= cur_t) && (flag[i]
== 0))                  {
                                minpri = priority[i];
                                c = i;
                }
                }
                if(st[c] == -1)
        {
                        st[c] = cur_t;
                ct[c] = cur_t;
        }
                b[c]--;
cur_t++;                        ct[c]++;
                for(i = 0; i < n; i++)
        {
                        if((c != i) && (st[i] != -1) && (flag[i] == 0))
                                ct[i]++;
        }
                if(b[c] == 0)
        {
                        flag[c] = 1;
                        tot++;
        }
                minpri = 1000;
```

```
                }
                int  tat = 0;                          //Turnaround Time
                int wt = 0;                                    //Waiting Time array
 float sum1 = 0, sum2 = 0, rt = 0;    //sum1 for tat and sum2 for wt and response time
                System.out.println("\nProcess
No.\tPriority\tA.T\tB.T.\tC.T.\tT.A.T.\tW.T.\tR.T.");
                for(i = 0; i < n; i++)
        {
                        tat = ct[i] - at[i];
                wt = tat - bt[i];
        rt = st[i] - at[i];
                        System.out.println("Process
"+p[i]+"\t"+priority[i]+"\t\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+tat+"\t"+wt+"\t"+rt);
                        sum1 += tat;
                        sum2 += wt;
        }
                System.out.println("Average Turn Around Time: "+(sum1/n));
                System.out.println("Average Waiting Time: "+(sum2/n));
                //To prepare Gantt Chart processes are soted according to start
time             for(i = 0; i < n; i++)                {
                        min1 = st[i];
                min2 = ct[i];
        min3 = p[i];
j = i-1;
                        while(j >= 0 && st[j] > min1)
                {
                                st[j+1] = st[j];
                ct[j+1] = ct[j];
p[j+1] = p[j];
                                j--;
                        }
                        st[j+1] = min1;
                ct[j+1] = min2;
                        p[j+1] = min3;
        }
 System.out.println("\n\t\tGANTT CHART\nPROCESS \tStart Time\tCompletion Time");
                for(i = 0; i < n; i++)
        {
                        System.out.println("PROCESS "+p[i]+"\t\t"+st[i]+"\t\t"+ct[i]);

                }
        }
```

}

```
Enter number of processes: 7
Enter arrival time: 0
Enter burst time: 4
Enter priority [smallest number is high]: 7
Enter arrival time: 1
Enter burst time: 2
Enter priority [smallest number is high]: 6
Enter arrival time: 2
Enter burst time: 3
Enter priority [smallest number is high]: 5
Enter arrival time: 3
Enter burst time: 5
Enter priority [smallest number is high]: 2
Enter arrival time: 4
Enter burst time: 1
Enter priority [smallest number is high]: 4
Enter arrival time: 5
Enter burst time: 4
Enter priority [smallest number is high]: 1
Enter arrival time: 6
Enter burst time: 6
Enter priority [smallest number is high]: 3

Process No.     Priority        A.T     B.T.    C.T.    T.A.T.  W.T.    R.T.
Process 1       7               0       4       25      25      21      0.0
Process 2       6               1       2       22      21      19      0.0
Process 3       5               2       3       21      19      16      0.0
Process 4       2               3       5       12      9       4       0.0
Process 5       4               4       1       19      15      14      14.0
Process 6       1               5       4       9       4       0       0.0
Process 7       3               6       6       18      12      6       6.0
Average Turn Around Time: 15.0
Average Waiting Time: 11.428572
```

```
                GANTT CHART
PROCESS         Start Time      Completion Time
PROCESS 1               0               25
PROCESS 2               1               22
PROCESS 3               2               21
PROCESS 4               3               12
PROCESS 6               5               9
PROCESS 7               12              18
PROCESS 5               18              19
```

4) Round Robin

```
import java.util.Scanner;
    class Queue
    {
            Node front, rear;
    int queueSize;
            class Node
            {               int
    data;
    Node next;
            }
            Queue()         {
            front = null;
```

```
rear = null;
queueSize = 0;
        }
        boolean isEmpty()
        {
                return (queueSize == 0);
        }
       int dequeue()
{
 int data = front.data;    front =
front.next;
                if (isEmpty())
        {
                        rear = null;
        }
                queueSize--;
        return data;
}
        void enqueue(int data)
        {
                Node oldRear = rear;
        rear = new Node();
rear.data = data;                rear.next =
null;
                if (isEmpty())
        {
                        front = rear;
                }
        else
                {
                        oldRear.next = rear;
                }
                queueSize++;
        }
}

class RR
{
        public static void main(String args[])
        {
 Scanner sc = new Scanner(System.in);   int n, tq;
                System.out.print("Enter number of processes: ");
```

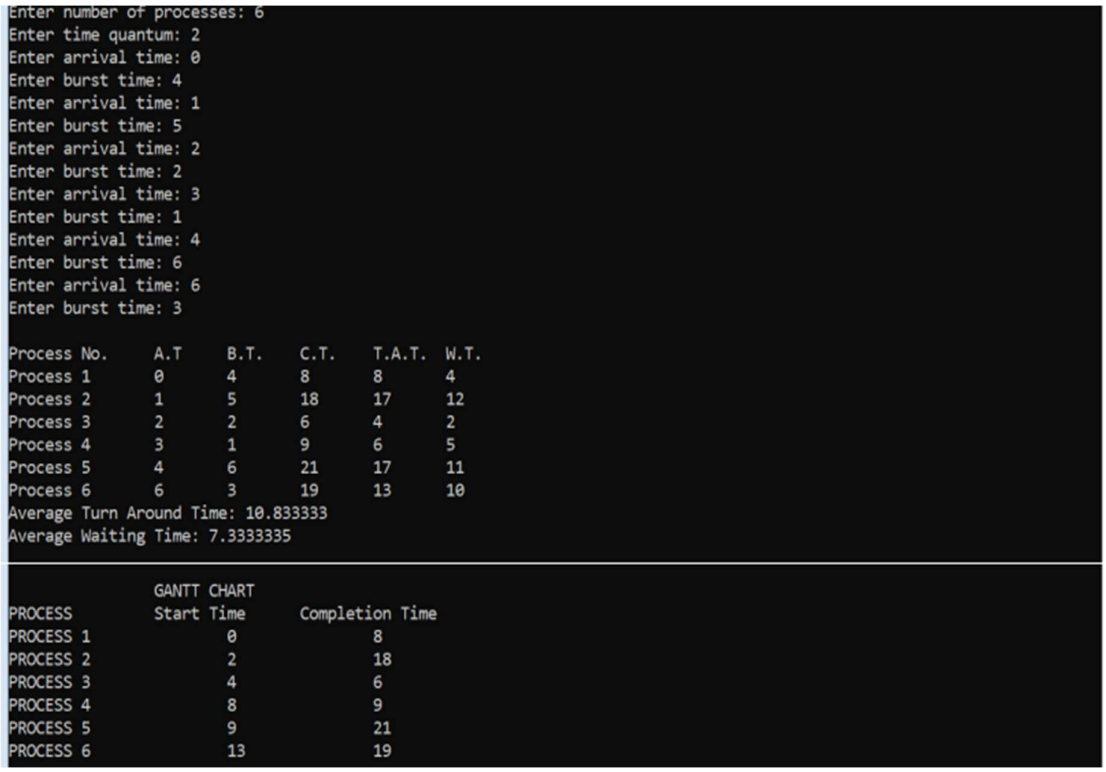
```
		n = sc.nextInt();
		System.out.print("Enter time quantum: ");
		tq = sc.nextInt();
		Queue q = new Queue();
		int p[] = new int[n];			//Process id array int
		at[] = new int[n];		//Arrival time array
		int bt[] = new int[n];			//Burst time array
		int flag[] = new int[n];		//To check if process is completed
		int f[] = new int[n];	int		//To check if process has arrived
		i, j, min1, min2, min3;
		int cur_t = 0;					//Current time
		int st[] = new int[n];			//Starting time of each process
		int ct[] = new int[n];			//completion time array
		int tot = 0;					//To count number of
processes completed
		int c = 0;					 //To track id of process to
be scheduled next
		int b[] = new int[n];			//Copy of Burst time array
		for(i = 0; i < n; i++)
		{
			p[i] = i+1;
			System.out.print("Enter arrival time: ");
			at[i] = sc.nextInt();
			System.out.print("Enter burst time: ");
			bt[i] = sc.nextInt();
		flag[i] = 0;
f[i] = 0;				st[i] = -1;
	}
		for(i = 0; i < n; i++)
	{
			min1 = at[i];
	min2 = bt[i];
min3 = p[i];
			j = i-1;
			while(j >= 0 && at[j] > min1)
	{
				at[j+1] = at[j];
			bt[j+1] = bt[j];
		p[j+1] = p[j];
				j--;
			}
			at[j+1] = min1;
```

```
                bt[j+1] = min2;
                p[j+1] = min3;
        }
        for(i = 0; i < n; i++)
                b[i] = bt[i];
        while(tot < n)
        {
                for(i = 0; i < n; i++)
                {
                        if((f[i] == 0) && (flag[i] == 0) && (at[i] <= cur_t))
                        {
                                q.enqueue(i);
                                f[i] = 1;
                        }
                }
                c = q.dequeue();
        if(st[c] == -1)
        st[c] = cur_t;                  ct[c]
= cur_t;                        if(b[c] > tq)
                {
                        ct[c] += tq;
        b[c] -= tq;
cur_t += tq;
                        for(i = 0; i < n; i++)
                {
                        if((f[i] == 0) && (flag[i] == 0) && (at[i] <= cur_t))
                        {
                                q.enqueue(i);
                        f[i] = 1;
                }
                        }
                        if(b[c] > 0)
                                q.enqueue(c);
        }
                else if(b[c] <= tq)
                {
                        ct[c] += b[c];
        cur_t += b[c];
b[c] = 0;
                        for(i = 0; i < n; i++)
                {
                        if((f[i] == 0) && (flag[i] == 0) && (at[i] <= cur_t))
```

```
                                        {
                                                q.enqueue(i);
                                                f[i] = 1;
                                        }
                                }
                        }
                        if(b[c] == 0)
                        {
                                flag[c] = 1;
                                tot++;
                        }
                }
                int  tat = 0;                                   //Turnaround Time int
                wt = 0;                                         //Waiting Time array
                float sum1 = 0, sum2 = 0;     //sum1 for tat and sum2 for wt
System.out.println("\nProcess No.\tA.T\tB.T.\tC.T.\tT.A.T.\tW.T.");   for(i = 0; i < n;
i++)   {
                        tat = ct[i] - at[i];
        wt = tat - bt[i];
System.out.println("Process
"+p[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+tat+"\t"+wt);
                        sum1 += tat;
                        sum2 += wt;
        }
                System.out.println("Average Turn Around Time: "+(sum1/n));
                System.out.println("Average Waiting Time: "+(sum2/n));

                //To prepare Gantt Chart processes are soted according to start time
        for(i = 0; i < n; i++)                {
                        min1 = st[i];
                min2 = ct[i];
        min3 = p[i];
j = i-1;
                        while(j >= 0 && st[j] > min1)
                {
                                st[j+1] = st[j];
                ct[j+1] = ct[j];
p[j+1] = p[j];
                                j--;
                        }
                        st[j+1] = min1;
                ct[j+1] = min2;
```

```
                        p[j+1] = min3;
        }
                System.out.println("\n\t\tGANTT CHART\nPROCESS \tStart
Time\tCompletion Time");
                for(i = 0; i < n; i++)
                {
                        System.out.println("PROCESS "+p[i]+"\t\t"+st[i]+"\t\t"+ct[i]);
                }
        }
}
```



Conclusion: Hence we have implemented various CPU scheduling algorithms (FCFS, SJF, Priority Scheduling and Round Robin).