Practical 5

1) Implement SJF (with no preemption) scheduling algorithm in java.

1. Imports and Class Definition:

```
package com.mycompany.sjf;
import java.util.*;
 public class Sjf {
```

- The code starts by defining the package and importing the necessary java.util package for using the Scanner class.
- The Sjf class is defined.

2. Main Method:

```
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
 int n;
```

- The main method is the entry point of the program.
- A Scanner object is created for reading user input.
- An integer variable n is declared to store the number of processes.

3. Matrix Declaration:

```
int[][] A = new int[100][4];
 int total = 0;
 float avg_wt, avg_tat;
```

- A 2D array A with 100 rows and 4 columns is declared to store process information. Each row will store information for one process: Process ID, Burst Time (BT), Waiting Time (WT), and Turn Around Time (TAT).
- total is used to accumulate the total waiting or turnaround time.
- avg_wt and avg_tat will store the average waiting and turnaround times.

4. User Input for Number of Processes:

```
System.out.println("Enter number of process:");
 n = input.nextInt();
```

- The user is prompted to enter the number of processes.
- The input is read and stored in n.

5. User Input for Burst Times:

```
System.out.println("Enter Burst Time:");
for (int i = 0; i < n; i++) {
 System.out.print("P" + (i + 1) + ": ");
A[i][1] = input.nextInt();
A[i][0] = i + 1;
 }
```

- The user is prompted to enter the burst time for each process.
- The burst time for each process is stored in the array A, and the process ID is also assigned.

6. Sorting Processes by Burst Time:

```
for (int i = 0; i < n; i++) {
    int index = i;
    for (int j = i + 1; j < n; j++) {
        if (A[j][1] < A[index][1]) {
            index = j;
        }
    }
    int temp = A[i][1];
    A[i][1] = A[index][1];
    A[index][1] = temp;
    temp = A[i][0];
    A[i][0] = A[index][0];
    A[index][0] = temp;
}
```

- The processes are sorted based on their burst time using a simple selection sort algorithm. The process with the shortest burst time is moved to the front.

7. Initialize Waiting Time for First Process:

```
A[0][2] = 0;
```

- The waiting time for the first process is set to 0 because it starts execution immediately.

8. Calculate Waiting Times:

```
for (int i = 1; i < n; i++) {
    A[i][2] = 0;
    for (int j = 0; j < i; j++) {
        A[i][2] += A[j][1];
    }
    total += A[i][2];
}
avg_wt = (float)total / n;
```

- For each process, the waiting time is calculated as the sum of the burst times of all previous processes.
- The total waiting time is accumulated, and the average waiting time is calculated.

9. Calculate Turnaround Times and Print Results:

```
total = 0;

System.out.println("P\tBT\tWT\tTAT");

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

    A[i][3] = A[i][1] + A[i][2];

    total += A[i][3];

    System.out.println("P" + A[i][0] + "\t"

                    + A[i][1] + "\t" + A[i][2]

                    + "\t" + A[i][3]);

}

avg_tat = (float)total / n;
```

- The turnaround time for each process is calculated as the sum of its burst time and waiting time.
- The total turnaround time is accumulated, and the results are printed in a tabular format.
- The average turnaround time is calculated.

10. Print Average Times:

```
System.out.println("Average Waiting Time= "
                + avg_wt);
System.out.println("Average Turnaround Time= "
                + avg_tat);
}
```

- The average waiting time and average turnaround time are printed.

Summary

The program simulates the Shortest Job First (SJF) scheduling algorithm by:

1. Reading the number of processes and their burst times.
2. Sorting the processes by burst time.
3. Calculating the waiting time and turnaround time for each process.
4. Printing the process details along with average waiting and turnaround times.

```
package com.mycompany.sjf;

import java.util.*;

public class Sjf {
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        int n;
        // Matrix for storing Process Id, Burst
        // Time, Average Waiting Time & Average
        // Turn Around Time.
        int[][] A = new int[100][4];
        int total = 0;
        float avg_wt, avg_tat;
        System.out.println("Enter number of process:");
        n = input.nextInt();
        System.out.println("Enter Burst Time:");
        for (int i = 0; i < n; i++) {
            // User Input Burst Time and alloting
```

```
    // Process Id.
    System.out.print("P" + (i + 1) + ": ");
    A[i][1] = input.nextInt();
    A[i][0] = i + 1;
}
for (int i = 0; i < n; i++) {
    // Sorting process according to their
    // Burst Time.
    int index = i;
    for (int j = i + 1; j < n; j++) {
        if (A[j][1] < A[index][1]) {
            index = j;
        }
    }
    int temp = A[i][1];
    A[i][1] = A[index][1];
    A[index][1] = temp;
    temp = A[i][0];
    A[i][0] = A[index][0];
    A[index][0] = temp;
}
A[0][2] = 0;
// Calculation of Waiting Times
for (int i = 1; i < n; i++) {
    A[i][2] = 0;
```

```
            for (int j = 0; j < i; j++) {
                A[i][2] += A[j][1];
            }
            total += A[i][2];
        }
        avg_wt = (float)total / n;
        total = 0;
        // Calculation of Turn Around Time and printing the
        // data.
        System.out.println("P\tBT\tWT\tTAT");
        for (int i = 0; i < n; i++) {
            A[i][3] = A[i][1] + A[i][2];
            total += A[i][3];
            System.out.println("P" + A[i][0] + "\t"
                               + A[i][1] + "\t" + A[i][2]
                               + "\t" + A[i][3]);
        }
        avg_tat = (float)total / n;
        System.out.println("Average Waiting Time= "
                           + avg_wt);
        System.out.println("Average Turnaround Time= "
                           + avg_tat);
    }
}
```