PROGRAM 2

2. Design, develop and implement program to simulate the working of Shortest Remaining Time First scheduling algorithm. Experiment with different length jobs.

SRTF Scheduling Algorithm

Input:

Read the number of processes n.

For each process i (from 1 to n), read:

arr[i]: Arrival time of the process.

bur[i]: Burst time (execution time) of the process.

Initialize:

rt[i]: Remaining time for each process, initially set to bur[i].

pr[i]: Process ID, set to i + 1 for each process.

comp: Number of completed processes, initialized to 0.

t: Current time, initialized to 0.

ttat: Total turnaround time, initialized to 0.

twt: Total waiting time, initialized to 0.

While comp < n (all processes are not completed):

Step 1: Select the next process to execute:

Set pos = -1 (to mark the process with the shortest remaining time).

For each process i (from 1 to n):

If arr[i] <= t (process has arrived) and rt[i] > 0 (process still has remaining time):

If pos == -1 (no process selected yet) or rt[i] < rt[pos] (current process has a shorter remaining time):

Set pos = i (select this process).

Step 2: Process Execution:

If pos == -1 (no process is available at the current time):

Increment t (move time forward) and continue to the next iteration.

Else:

Decrease the remaining time of the selected process: rt[pos]--.

Increment the current time: t++.

Step 3: Process Completion:

If rt[pos] == 0 (the process is completed):

Increment the number of completed processes: comp++.

Set the finish time of the process: ft[pos] = t.

Calculate the turnaround time: tat[pos] = ft[pos] - arr[pos].

Calculate the waiting time: wt[pos] = tat[pos] - bur[pos].

After all processes are completed:

Calculate the total turnaround time (ttat) and total waiting time (twt) by summing up the respective values for all processes.

Calculate the average turnaround time: avgtat = ttat / n.

Calculate the average waiting time: avgwt = twt / n.

Output:

For each process i (from 1 to n), output:

Process ID, Burst Time, Arrival Time, Turnaround Time, and Waiting Time.

Output the average turnaround time (avgtat) and average waiting time (avgwt).

Program:

```
#include <stdio.h>

int arr[10], bur[10], rt[10], n, pr[10], t = 0, ft[10], wt[10], tat[10], pos, total = 0, ttat = 0, twt = 0;
float avgtat, avgwt;

int main()
{
   int comp = 0, i;

   // Input for number of processes
   printf("\nEnter the number of processes: ");
   scanf("%d", &n);

   // Input for arrival time and burst time for each process
   for (i = 0; i < n; i++)
 {
      printf("\nEnter the arrival and burst time for process %d: ", i + 1);
      scanf("%d %d", &arr[i], &bur[i]);
      pr[i] = i + 1; // Process ID
   }

   // Initialize remaining time (rt) with burst time (bur)
   for (i = 0; i < n; i++)
      rt[i] = bur[i];
```

```
    // Main loop to simulate SRTF
    while (comp < n)
 {
        // Finding the process with shortest remaining time that has arrived
        pos = -1; // Reset pos before each loop

        for (i = 0; i < n; i++)
{
            if (arr[i] <= t && rt[i] > 0)
{ // Process must have arrived and still have burst time left
                if (pos == -1 || rt[i] < rt[pos])
{ // First eligible process or shorter burst time
                    pos = i;
                }
            }
        }

        // If no process found, move time forward
        if (pos == -1)
{
            t++;
            continue;
        }

        // Decrease the remaining time for the selected process
        rt[pos]--;
        t++;
```

```
        // If process is completed
        if (rt[pos] == 0)
{
            comp++; // Completed process count
            ft[pos] = t; // Finish time of the process
            tat[pos] = ft[pos] - arr[pos]; // Turnaround time = Finish time - Arrival time
            wt[pos] = tat[pos] - bur[pos]; // Waiting time = Turnaround time - Burst time
        }
    }

    // Calculate total TAT and WT for average calculation
    for (i = 0; i < n; i++)
{
        ttat += tat[i];
        twt += wt[i];
    }

    // Calculate averages
    avgtat = (float)ttat / n;
    avgwt = (float)twt / n;

    // Output the process details and average times
    printf("\nProcess\tBT\tAT\tTAT\tWT\n");
    for (i = 0; i < n; i++)
 {
        printf("%d\t%d\t%d\t%d\t%d\n", pr[i], bur[i], arr[i], tat[i], wt[i]);
```

```
    }

    printf("Avg TAT = %.2f\nAvg WT = %.2f\n", avgtat, avgwt);

    return 0;
}
```