Experiment no: 5

Name: Saloni Vishwakarma
Batch-Roll no: C1-13
Semester-Section:4th-C
Subject: OS(Operating System) Lab
Date of execution: 17 May 2023

Aim: To write and execute C programs to demonstrate different CPU scheduling algorithms.
5B) To write and execute C programs to demonstrate Shortest Job First pre-emptive CPU scheduling algorithm.

Theory:

Shortest Job First (SJF) with preemption is a CPU scheduling algorithm where the process with the shortest burst time is given the highest priority. If a new process arrives with a shorter burst time than the one currently running, the running process is preempted, and the new process is scheduled to run.

Here's an overview of the SJF algorithm with preemption:

1. Start with an empty ready queue and a running process (if any).
2. When a process arrives, compare its burst time with the remaining burst time of the running process (if any).
3. If the new process has a shorter burst time, preempt the running process and add it to the ready queue.
4. Schedule the new process to run.
5. If the new process finishes before any other process arrives, continue to the next step.
6. If a new process arrives during the execution of the current process and has a shorter burst time, preempt the current process and add it to the ready queue.
7. Schedule the new process to run.
8. Repeat steps 5 to 7 until all processes have completed.

The SJF algorithm with preemption ensures that the process with the shortest burst time always gets the CPU, even if a new process with an even shorter burst time arrives later. This can help minimize the average waiting time and improve the overall turnaround time for processes.

It's worth noting that SJF with preemption can lead to starvation for processes with long burst times. If there are always shorter processes arriving, the longer processes may never get a chance to run. To mitigate this issue, a time-based priority boost or other techniques can be applied to ensure fairness among processes.

Code and Output:

```
#include <stdio.h>
int main()
{
   int arrival_time[10], burst_time[10], temp[10];
   int i, smallest, count = 0, time, limit;
   double wait_time = 0, turnaround_time = 0, end;
   float average_waiting_time, average_turnaround_time;
   printf("\n Enter the total number of processes: ");
   scanf("%d", &limit);
   printf("\n Enter details of %d processes", limit);
   for(i = 0; i < limit; i++)
   {
      printf("\n Enter arrival time for process %d: ",i+1);
      scanf("%d", &arrival_time[i]);
      printf(" Enter burst time for process %d: ",i+1);
      scanf("%d", &burst_time[i]);
      temp[i] = burst_time[i];
   }
   printf("\n *******SHORTEST JOB FIRST WITH PRE-EMPTION*******\n");
   printf("\n Process\tTurn Around Time\tWaiting Time\n" );
   burst_time[9] = 9999;
   for(time = 0; count != limit; time++)
   {
      smallest = 9;
      for(i = 0; i < limit; i++)
      {
         if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] && burst_time[i]
> 0)
         {
            smallest = i;
         }
```

```
        }
        burst_time[smallest]--;
        if(burst_time[smallest] == 0)
        {
            count++;
            end = time + 1;
            printf("\n P[%d]\t|\t %lf\t|\t
%lf",smallest+1,end-arrival_time[smallest],end-arrival_time[smallest]-temp[smallest]);
            wait_time = wait_time + end - arrival_time[smallest] -
            temp[smallest];
            turnaround_time = turnaround_time + end - arrival_time[smallest];
        }
    }
    printf("\n");
    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;
    printf("\n Average Waiting Time= %lf\n", average_waiting_time);
    printf("\n Average Turnaround Time= %lf\n", average_turnaround_time);
    return 0;
}
```

```
Enter the total number of processes: 4

Enter details of 4 processes
Enter arrival time for process 1: 0
Enter burst time for process 1: 5

Enter arrival time for process 2: 1
Enter burst time for process 2: 3

Enter arrival time for process 3: 2
Enter burst time for process 3: 4

Enter arrival time for process 4: 4
Enter burst time for process 4: 1

*******SHORTEST JOB FIRST WITH PRE-EMPTION*******

Process        Turn Around Time        Waiting Time

P[2]   |        3.000000       |        0.000000
P[4]   |        1.000000       |        0.000000
P[1]   |        9.000000       |        4.000000
P[3]   |        11.000000      |        7.000000

Average Waiting Time= 2.750000

Average Turnaround Time= 6.000000
```