Write program to implement SJF scheduling algorithm

CODE

```
//SJF scheduling algorithm

#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Please enter the number of processes: ";
    cin>>n;

    int burst[n],process[n];

    for(int i=0; i<n; i++)
    {
        cout<<"Please enter the CPU Burst Time for process P"<<i+1<<" :";
        cin>>burst[i];
        process[i]=i+1;
    }

    int j,k;
    int temp1,temp2;
    for(j=1; j<n; j++)                  //sorting burst time and swapping elements
of process[] along with burst[]
    {
        temp1=burst[j];
        temp2=process[j];
        for(k=j; k>0 && temp1<burst[k-1]; k--)
        {
            burst[k]=burst[k-1];
            process[k]=process[k-1];
        }

        burst[k]=temp1;
        process[k]=temp2;
    }
```

```
    int wait_time[n],turnaround_time[n];

    wait_time[0]=0;
    for(int i=1; i<n; i++)
        wait_time[i]=wait_time[i-1]+burst[i-1];     //calculating wait time

    for(int i=0; i<n; i++)
        turnaround_time[i]=wait_time[i]+burst[i];       //calculating turnaround
time



    float avg_wt=0,avg_tt=0;

    cout<<"Processes \t  Burst Time \t   Waiting Time  \t  Turnaround
Time"<<endl;        //printing wait time & turnaround time
    for(int i=0; i<n; i++)
    {
        cout<<"P"<<process[i]<<"  \t\t   "<<burst[i]<<"   \t\t      "<<wait_time
[i]<<"    \t\t\t   "<<turnaround_time[i]<<endl;
        avg_wt+=wait_time[i];               //calculating total sum of wait time
        avg_tt+=turnaround_time[i];         //calculating total sum of
turnaround time
    }

    avg_wt=avg_wt/n;                //calculating avg wait time
    avg_tt=avg_tt/n;                //calculating avg turnaround time

    cout<<"\nAverage Waiting Time = "<<avg_wt<<endl;
    cout<<"\nAverage Turnaround Time = "<<avg_tt<<endl;

    return 0;
}
```

OUTPUT

```
➜ OSPracticals g++ Practical8.cpp -o Practical8
➜ OSPracticals ./Practical8
Please enter the number of processes: 4
Please enter the CPU Burst Time for process P1 :21
Please enter the CPU Burst Time for process P2 :69
Please enter the CPU Burst Time for process P3 :42
Please enter the CPU Burst Time for process P4 :22
Processes         Burst Time       Waiting Time          Turnaround Time
P1                 21                          0                          21
P4                 22                          21                         43
P3                 42                          43                         85
P2                 69                          85                         154

Average Waiting Time = 37.25

Average Turnaround Time = 75.75
➜ OSPracticals
```

Write program to implement SRTF scheduling algorithm

CODE

```
#include <iostream>

using namespace std;

void waiting_time(struct process a[], int n);

struct process
{

    int process_id;
    int burst_time;
    int waiting_time;
    int arrival_time;
    int remain_time;

} arr[100];

int process_finish[100];

int main()
{
    arr[99].remain_time = 9999;

    int n; //No of process in variable n

    cout << "\nPlease enter the number of Processes : ";
    cin >> n;
    cout << endl;

    for (int i = 0; i < n; i++) //Take the Burst time for each process by using
loop
    {

        arr[i].process_id = i + 1; //increment the process_id by 1 after each
burst_time
```

```
        cout << "Please enter the CPU Burst Time of P" << i + 1 << " : ";

        cin >> arr[i].burst_time;

        arr[i].remain_time = arr[i].burst_time; //copy each process burst_time
to another array remain_time[]

        cout << "Please enter the Arrival Time : ";

        cin >> arr[i].arrival_time;
        cout << endl;
    }

    waiting_time(arr, n);
    return 0;
}

void waiting_time(struct process a[], int n)

{

    int remain = 0, sum_wait = 0, sum_turnaround = 0, endTime, smallest;

    cout << "\n\nProcess  Turnaround Time  Waiting Time\n\n";

    int process_f = 0; // handle the INDEX of array process_finish.

    for (int time = 0; remain != n; time++)
    {
        smallest = 99;

        for (int i = 0; i < n; i++)
        {
            if ((a[i].arrival_time <= time) && (a[i].remain_time <
a[smallest].remain_time) && (a[i].remain_time > 0))
            {
                smallest = i;
            }
        }
```

```
        a[smallest].remain_time--;

        if (a[smallest].remain_time == 0)
        {
            process_finish[process_f] = smallest + 1; //to assign a process # which finish the total job
            process_f++;
            a[smallest].process_id = smallest + 1; //to ssign a process_id

            int tt;

            remain++; //One process complete the total job

            endTime = time + 1; //Total competional time of process

            tt = endTime - a[smallest].arrival_time; //Calculate the TURNaround TIME (competionalTime - TT )

            a[smallest].waiting_time = tt - a[smallest].burst_time; //Calculate the Waiting Time

            cout << "\nP[" << smallest + 1 << "]\t\t" << tt << "\t\t" << a[smallest].waiting_time;

            sum_wait += tt - a[smallest].burst_time; //For find Average Waiting Time
        }
    }

    cout << "\n\nAverage Waiting Time = " << sum_wait * 1.0 / n;
}
```

OUTPUT

```
➜ OSPracticals g++ Practical11.cpp -o Practical11
➜ OSPracticals ./Practical11

Please enter the number of Processes : 3

Please enter the CPU Burst Time of P1 : 10
Please enter the Arrival Time : 1

Please enter the CPU Burst Time of P2 : 20
Please enter the Arrival Time : 2

Please enter the CPU Burst Time of P3 : 30
Please enter the Arrival Time : 3



Process  Turnaround Time  Waiting Time


P[1]            10              0
P[2]            29              9
P[3]            58              28

Average Waiting Time = 12.3333%
➜ OSPracticals
```