	SVKM's NMIMS School of Technology Management & Engineering Navi Mumbai Campus
	Department of Computer Engineering

Name: Rahul Purohit	**SAP ID: 70562100106**
Semester: III	**Year: II**
Subject: Operating systems	**Roll No.: A022**
Batch: 2	**Date:**

Shortest Job First (SJF):

- SJF is a non-preemptive scheduling algorithm, meaning once a process starts executing, it continues until it finishes.
- The scheduler selects the process with the shortest burst time to execute next.
- SJF minimizes the average waiting time for processes because it prioritizes shorter jobs.
- It can suffer from the "starvation" problem, where long jobs may never get CPU time if short jobs keep arriving.
- SJF can be either preemptive (select the shortest job from the ready queue at each scheduling point) or non-preemptive (select the shortest job from the entire queue when a process finishes or a new one arrives).

Shortest Remaining Time First (SRTF):

- SRTF is a preemptive version of SJF. It selects the process with the shortest remaining burst time to execute next.
- SRTF dynamically adapts to the arrival of new processes or changes in the remaining time of running processes.
- It can provide optimal turnaround times, as it always selects the shortest remaining time process.
- SRTF avoids the starvation problem by periodically reassessing the ready queue and selecting the shortest job.
- While it minimizes waiting times and provides fairness, it requires frequent context switching, which can lead to higher overhead.
- SRTF can be challenging to implement efficiently because it requires constant monitoring and preemption.

SJF Code:

```cpp
#include<iostream>
#include <algorithm>
#include<iomanip>
#include<climits>
using namespace std;

struct process_struct
{
  int pid;
  int at;
  int bt;
  int ct,wt,tat,rt,start_time;
}ps[100];

int main()
{

    int n;
    bool is_completed[100]={false},is_first_process=true;
    int current_time = 0;
    int completed = 0;;
    cout<<"Enter total number of processes: ";
    cin>>n;
    int sum_tat=0,sum_wt=0,sum_rt=0,total_idle_time=0,prev=0,length_cycle;
    float cpu_utilization;
    int max_completion_time,min_arrival_time;

    cout << fixed << setprecision(2);

    for(int i=0;i<n;i++)
    {
        cout<<"\nEnter Process " <<i<< " Arrival Time: ";
        cin >> ps[i].at;
        ps[i].pid=i;
    }

    for(int i=0;i<n;i++)
    {
        cout<<"\nEnter Process " <<i<< " Burst Time: ";
        cin >> ps[i].bt;
    }


    while(completed!=n)
    {
        int min_index = -1;
        int minimum = INT_MAX;
        for(int i = 0; i < n; i++) {
            if(ps[i].at <= current_time && is_completed[i] == false) {
                if(ps[i].bt < minimum) {
                    minimum = ps[i].bt;
                    min_index = i;
                }
                if(ps[i].bt== minimum) {
                    if(ps[i].at < ps[min_index].at) {
                        minimum= ps[i].bt;
                        min_index = i;
                    }
                }
            }
        }

        if(min_index==-1)
        {
            current_time++;
        }
        else
        {
        ps[min_index].start_time = current_time;
        ps[min_index].ct = ps[min_index].start_time + ps[min_index].bt;
        ps[min_index].tat = ps[min_index].ct - ps[min_index].at;
        ps[min_index].wt = ps[min_index].tat - ps[min_index].bt;
        ps[min_index].rt = ps[min_index].wt;

        sum_tat +=ps[min_index].tat;
        sum_wt += ps[min_index].wt;
        sum_rt += ps[min_index].rt;
        total_idle_time += (is_first_process==true) ? 0 : (ps[min_index].start_time -  prev);

        completed++;
        is_completed[min_index]=true;
        current_time = ps[min_index].ct;
        prev= current_time;
        is_first_process = false;
        }
    }
    cout<<"\nProcess No.\tAT\tCPU Burst Time\tCT\tTAT\tWT\n";
    for(int i=0;i<n;i++)
     cout<<i<<"\t\t"<<ps[i].at<<"\t"<<ps[i].bt<<"\t\t"<<ps[i].ct<<"\t"<<ps[i].tat<<"\t"<<ps[i].wt<<endl;
    cout<<endl;
    cout<<"\nAverage Turn Around time= "<< (float)sum_tat/n;
    cout<<"\nAverage Waiting Time= "<<(float)sum_wt/n;
    cout<<"\nAverage Response Time= "<<(float)sum_rt/n;
    return 0;
}
```

Output:

```
Enter total number of processes: 5

Enter Process 0 Arrival Time: 1

Enter Process 1 Arrival Time: 2

Enter Process 2 Arrival Time: 3

Enter Process 3 Arrival Time: 1

Enter Process 4 Arrival Time: 4

Enter Process 0 Burst Time: 3

Enter Process 1 Burst Time: 2

Enter Process 2 Burst Time: 5

Enter Process 3 Burst Time: 3

Enter Process 4 Burst Time: 1

Process No.     AT      CPU Burst Time  CT      TAT     WT
0               1       3               4       3       0
1               2       2               7       5       3
2               3       5               15      12      7
3               1       3               10      9       6
4               4       1               5       1       0


Average Turn Around time= 6.00
Average Waiting Time= 3.20
Average Response Time= 3.20
```

SRTF CODE:

```cpp
#include<iostream>
#include<algorithm>
#include<iomanip>
#include<climits>
using namespace std;

struct process_struct
{
  int pid;
  int at;
  int bt;
  int ct,wt,tat,rt,start_time;
}ps[100];


int main()
{

    int n;
    float bt_remaining[100];
    bool is_completed[100]={false},is_first_process=true;
    int current_time = 0;
    int completed = 0;;
    float sum_tat=0,sum_wt=0,sum_rt=0,total_idle_time=0,length_cycle,prev=0;
    float cpu_utilization;
    int max_completion_time,min_arrival_time;

    cout << fixed << setprecision(2);

    cout<<"Enter total number of processes: ";
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cout<<"\nEnter Process" <<i<< "Arrival Time: ";
        cin >> ps[i].at;
        ps[i].pid=i;
    }

    for(int i=0;i<n;i++)
    {
        cout<<"\nEnter Process" <<i<< "Burst Time: ";
        cin >> ps[i].bt;
        bt_remaining[i]= ps[i].bt;
    }


    while(completed!=n)
    {
        int min_index = -1;
        int minimum = INT_MAX;
        for(int i = 0; i < n; i++) {
            if(ps[i].at <= current_time && is_completed[i] == false) {
                if(bt_remaining[i] < minimum) {
                    minimum = bt_remaining[i];;
                    min_index = i;
                }
                if(bt_remaining[i]== minimum) {
                    if(ps[i].at < ps[min_index].at) {
                        minimum= bt_remaining[i];;
                        min_index = i;
                    }
                }
            }
        }


        if(min_index==-1)
        {
            current_time++;
        }
        else
        {
            if(bt_remaining[min_index] == ps[min_index].bt)
            {
                        ps[min_index].start_time = current_time;
                        total_idle_time += (is_first_process==true) ? 0 : (ps[min_index].start_time -  prev);
                        is_first_process=false;
            }
            bt_remaining[min_index] -= 1;
            current_time++;
            prev=current_time;
            if(bt_remaining[min_index] == 0)
            {
                ps[min_index].ct = current_time;
                ps[min_index].tat = ps[min_index].ct - ps[min_index].at;
                ps[min_index].wt= ps[min_index].tat - ps[min_index].bt;
                ps[min_index].rt = ps[min_index].start_time - ps[min_index].at;


                sum_tat +=ps[min_index].tat;
                sum_wt += ps[min_index].wt;
                sum_rt += ps[min_index].rt;
                completed++;
                is_completed[min_index]=true;


            }
        }
    }

    cout<<"\nProcess No.\tAT\tCPU Burst Time\tCT\tTAT\tWT\n";
    for(int i=0;i<n;i++)
     cout<<i<<"\t\t"<<ps[i].at<<"\t"<<ps[i].bt<<"\t\t"<<ps[i].ct<<"\t"<<ps[i].tat<<"\t"<<ps[i].wt<<endl;
    cout<<endl;


    cout<<"\nAverage Turn Around time= "<< (float)sum_tat/n;
    cout<<"\nAverage Waiting Time= "<<(float)sum_wt/n;
    cout<<"\nAverage Response Time= "<<(float)sum_rt/n;

    return 0;
}
```

Output:

```
Enter total number of processes: 5

Enter Process0Arrival Time: 1

Enter Process1Arrival Time: 2

Enter Process2Arrival Time: 4

Enter Process3Arrival Time: 3

Enter Process4Arrival Time: 2

Enter Process0Burst Time: 4

Enter Process1Burst Time: 3

Enter Process2Burst Time: 2

Enter Process3Burst Time: 1

Enter Process4Burst Time: 5

Process No.     AT      CPU Burst Time  CT      TAT     WT
0               1       4               6       5       1
1               2       3               11      9       6
2               4       2               8       4       2
3               3       1               4       1       0
4               2       5               16      14      9


Average Turn Around time= 6.60
Average Waiting Time= 3.60
Average Response Time= 3.40
```