SJF

```
#include <stdio.h>
struct process
{
	int at;
	int st;
	int status;
	int ft;
	int tt;
}ready_list[10];
int n,completed=0;
int main()
{
	int i,cur_time,pid,tim_slice;
	printf("Enter number of processes:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Process %d\n",i+1);
		printf("***********\n");
		printf("Enter Arrival Time:");
		scanf("%d",&ready_list[i].at);
		printf("Enter Service Time:");
		scanf("%d",&ready_list[i].st);
		ready_list[i].status=0;
	}
	i=0; cur_time=0;
	do{
	    pid=dispatcher(cur_time);
			while(pid ==-1 && completed != n) // Next process not yet arrived
			{
				cur_time++;
				pid=dispatcher(cur_time);
			 }
	    cur_time+=ready_list[pid].st; // Update current time
	    ready_list[pid].st = 0; // Current process completed
	    ready_list[pid].ft=cur_time; //Finish time
	    ready_list[pid].status=1; // Process status changed to completed
	    completed++;
	}while(completed <= n);
	for(i=0;i<n;i++)
	{
		ready_list[i].tt=ready_list[i].ft-ready_list[i].at;
	}
	printf("PID\t AT     ST        FT    TT\n");
```

```
        printf("***\t **     **    **    **\n");
        for(i=0;i<n;i++)
        {
        printf("%d\t%d\t%d\t%d\t%d\n",i,ready_list[i].at,ready_list[i].st,ready_list[i].ft,ready_list[i].tt);
        }
        getch();
}
int dispatcher(int time) //choose the shortest process among the arrived processes
{
   int ST=1000,index=-1,i;
     for(i=0;i<n;i++)
     {
        if(ready_list[i].status != 1)
        {
            if(ready_list[i].at <= time)
            {
               if(ready_list[i].st < ST)
               {
                  ST=ready_list[i].st;
                  index=i;
               }
            }
            }
     }
     return index;
}
```

SRT

```
#include <stdio.h>
struct process
{
        int at;
        int st;
        int status;
        int ft;
        int tt;
}ready_list[10];
int n,completed=0;
int dispatcher(int);
int main()
{
        int i,cur_time,pid,tim_slice;
        printf("Enter number of processes:");
        scanf("%d",&n);
```

```
        for(i=0;i<n;i++)
        {
                printf("Process %d\n",i+1);
                printf("***********\n");
                printf("Enter Arrival Time:");
                scanf("%d",&ready_list[i].at);
                printf("Enter Service Time:");
                scanf("%d",&ready_list[i].st);
                ready_list[i].status=0;
        }
        i=0; cur_time=0;
        do{
                pid=dispatcher(cur_time);
                while(pid ==-1 && completed != n)
                 {
                        cur_time++;
                        pid=dispatcher(cur_time);
                }
                cur_time+=1; // Allow the process to run one unit of time
                ready_list[pid].st--; // Reduce service time by one
                if(ready_list[pid].st==0) // If process has completed
                {
                        ready_list[pid].ft=cur_time;
                        ready_list[pid].status=1;
                        completed++;
                }
        }while(completed < n);
        for(i=0;i<n;i++)
        {
                ready_list[i].tt=ready_list[i].ft-ready_list[i].at;
        }
        printf("PID\t AT     ST         FT    TT\n");
        printf("***\t **     **    **    **\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t  %d\t%d\t%d\t%d\n",i,ready_list[i].at,ready_list[i].st,ready_list[i].ft,ready_list[i].tt);
        }
}
int dispatcher(int time)
{
   int ST=1000,index=-1,i;
     for(i=0;i<n;i++)
     {
                if(ready_list[i].status != 1)
                {
                        if(ready_list[i].at <= time)
```

```
                {
                        if(ready_list[i].st < ST)
                        {
                                ST=ready_list[i].st;
                                index=i;
                        }
                }
        }
    }
    return index;
}
```