```c
//Program for FCFS
#include <stdio.h>
struct process
{
    int at; // arrival time
    int bt; // burst time
    int status; // completed -1, not yet completed - 0
    int ft; // finish time
    int tt; // turn around time
    int wt; // wait time
}ready_list[10];
int n, cur_time=0, idle_time=0;
int dispatcher();
int main()
{
        int i,pid;
        printf("Enter number of processes:");
        scanf("%d",&n);
        for(i=0;i<n;i++) // To get process details
        {
            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].bt);
            ready_list[i].status=0;
        }
        i=0;
        while(i < n) // until all the processes are finished
        {
            pid=dispatcher(); // To identify the next process to be scheduled
            ready_list[pid].ft=cur_time + ready_list[pid].bt; // Finish time calculated
            ready_list[pid].status=1; // To mark that the process is already completed
            ready_list[pid].tt = ready_list[pid].ft - ready_list[pid].at;
            ready_list[pid].wt = ready_list[pid].tt - ready_list[pid].bt;
            cur_time = cur_time + ready_list[pid].bt; // To update the system clock
            i++;// one more process finished
        }
        printf("PId\tFinish Time\tTT\t\tWT\n");
        printf("****************************************\n");
        for(i=0;i<n;i++)
        {
        printf("%d\t%d\t\t%d\t\t%d\n",i,ready_list[i].ft, ready_list[i].tt,ready_list[i].wt);
        }
        printf("Total CPU idle time: %d", idle_time); // total time that CPU was idle
}
int dispatcher() // Function to pick the next process that arrived first
{

    int i,index=-1, shortBT=999;
    printf("Cur_time:%d", cur_time);
    back:
    for(i=0;i<n;i++)
    {
        if(ready_list[i].status != 1) // To check that ith process is not yet completed
            if(ready_list[i].at <= cur_time) // To check that ith process has arrived
            {
                if(ready_list[i].bt < shortBT)//Whether ith procees has shorter burst time
                {
```

```
                    index=i; //index of the process that is currently chosen
                    shortBT=ready_list[i].bt;
                }
            }
        }
        if(index == -1) // Next process not yet available at the current time
        {
            cur_time++; // To move the clock until it reach the arrival time of next process
            idle_time++; // Since CPU has been idle waiting for next process
            goto back;
        }
        printf("Index: %d\n", index);
        return index;
}
```