```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

#define MAX 10

typedef struct process
{
 char name[20];
 int at,bt,wt,ct,tat,tbt;  //tbt : temp burst time
}process;

process p[MAX];

int processcount,currenttime;
float totaltat,totalwt;
float avgtat,avgwt;

void sort()
{
 int i,j;
 process t;

 for(i=0;i<processcount;i++)
    {
     for(j=i+1;j<processcount;j++)
        {
         if(p[i].at > p[j].at)
          {
           t=p[i];
           p[i]=p[j];
           p[j]=t;
          }
        }
    }
}

void readprocess()
{
 int i;
 printf("\nEnter number of processes:");
 scanf("%d",&processcount);
 srand(time(NULL));
 for(i=0;i<processcount;i++)
    {
     printf("\nEnter process name:");
     scanf("%s",p[i].name);

     //printf("\nEnter process arrival time:");
     //scanf("%d",&p[i].at);
     p[i].at=rand()%10; //set range according to you

     //printf("\nEnter process burst time:");
     //scanf("%d",&p[i].bt);
     do{
        p[i].bt=rand()%10; //set range according to you
     }while(p[i].bt==0);  //burst time shouldn't be zero
```

```
     p[i].tbt=p[i].bt; //extra line from fcfs
    }
  sort();
}

int getprocess()
{
 int i=0,min=999,p1=-1;

 for(i=0;i<processcount;i++)
    {
     if(p[i].at<=currenttime && p[i].tbt!=0)
      {
        if(p[i].bt<min)  //compare burst time
          {
           min=p[i].bt;
           p1=i;
          }
      }
    }
 return p1;
}

void scheduleprocess()
{
 int i,cnt=0;
 printf("\n*******Gantt Chart**********\n");
 while(1)
   {
    i=getprocess(); //extra line from fcfs
    if(i==-1) //when cpu is idle
      {
       printf("%d idle ",currenttime);
       currenttime=p[cnt].at; //modified line from fcfs
       printf("%d|",currenttime);
      }
    else
      {
        printf("%d ",currenttime);
        p[i].wt=currenttime-p[i].at;
        currenttime=currenttime+p[i].bt;
        p[i].ct=currenttime;
        p[i].tat=p[i].wt+p[i].bt; //p[i].tat=p[i].ct-p[i].at;

        totaltat=totaltat+p[i].tat; //totaltat=totaltat+p[i].tat;
        totalwt=totalwt+p[i].wt;

        p[i].tbt=0; //extra line from fcfs

        printf("%s ",p[i].name);
        printf("%d|",currenttime);

        cnt++;
        if(cnt==processcount)
          break;
        }
```

```
      }

   avgtat=totaltat/processcount;
   avgwt=totalwt/processcount;
}

void displayprocess()
{
  int i;
  printf("\nName\tArrival\tBurst\tWaiting\tCPU time\tTurnAround");
  for(i=0;i<processcount;i++)
     {
      printf("\n%s\t%d\t%d\t%d\t%d\t\t%d",p[i].name,p[i].at,p[i].bt,p[i].wt,p[i].ct,p[i].tat
      );
     }
  printf("\nTotal Turnaround time %f",totaltat);
  printf("\nTotal Waiting time %f",totalwt);
  printf("\nAverage Turnaround time %f",avgtat);
  printf("\nAverage Waiting time %f",avgwt);
}

int main()
{
 readprocess();
 scheduleprocess();
 displayprocess();
 return 0;
}
```