```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

typedef struct PROCESS
{
 char name[80] ;
 int at,bt,ct,wt,tat,tbt;
}PROCESS ;

PROCESS p[10] ;
int processCount,q,tq,x;
float totaltat,totalwt,avgtat,avgwt ;

void sort()
{
 PROCESS p1 ;
 int i,j;
 for(i=0;i<processCount;i++)
    {
     for(j=i+1;j<processCount;j++)
        {
         if(p[j].at < p[i].at)
            {
             p1 = p[i] ;
             p[i] = p[j] ;
             p[j] = p1 ;
            }
        }
    }
}


void readProcess()
{
 int i ;
 printf("\nEnter the number of processes: ") ;
 scanf("%d",&processCount) ;
 printf("\nEnter time quantum:");
 scanf("%d",&q);
 for(i=0;i<processCount;i++)
    {
     printf("\nEnter the process name: ") ;
     scanf("%s",p[i].name) ;
     printf("Enter the CPU Burst time: ") ;
     scanf("%d",&p[i].bt) ;
     printf("Enter the Arrival time: ") ;
     scanf("%d",&p[i].at) ;
     p[i].tbt = p[i].bt ;
    }
 sort() ;
}
```

```
int getProcess()
{
 int p1;
 if(x==processCount)
   x=0;

 p1=x;

 x++;

 return p1 ;
}

void scheduleProcess()
{
 int i,count=0,time=0 ;
 char currentprocess[10] , prevprocess[10] = "NULL" ;
 printf("\n\n  GanttChart:\n") ;
 printf("_____________________________________________________\n") ;
 while(1)
      {
        tq=0;
        i=getProcess();

        if(p[i].tbt==0) //if i is finished
           tq=q;             //then dont iterate

        while(tq!=q)  //iterate quantum times
            {
            p[i].tbt-- ;
            strcpy(currentprocess,p[i].name) ;
            if(strcmp(currentprocess,prevprocess) !=0 )
              {
               printf("%d|%d  %s  ",time , time , currentprocess) ;
              }
            time++ ;
            if(p[i].tbt==0)
              {
               p[i].ct = time ;
               p[i].tat = time - p[i].at ;

               p[i].wt = p[i].tat-p[i].bt ;

               count++ ;
               totaltat += p[i].tat ;
               totalwt+=p[i].wt ;
              }
            strcpy(prevprocess , currentprocess) ;

            if(p[i].tbt==0 && tq!=q)  //if process is finished before time
            quantum
              break;
```

```
            tq++;
           }
        if(count==processCount)
            break ;

          }
 printf("%d|",time) ;
 printf("\n_____________________________________________________\n") ;
 avgtat = totaltat/processCount ;
 avgwt = totalwt/processCount ;
}

void display()
{
 int i;

 printf("\n-----------------------------------------------------------------------
 ------------\n") ;
 printf("Process  ArrivalTime  BurstTime  CPUTime  TurnAroundtime
 WaitTime\n");

 printf("-----------------------------------------------------------------------
 ----------\n");
 for(i=0 ; i<processCount ; i++)
    printf("%s\t  %d\t\t%d\t  %d\t   %d\t\t
    %d\n",p[i].name,p[i].at,p[i].bt,p[i].ct,p[i].tat,p[i].wt) ;

 printf("-----------------------------------------------------------------------
 ----------") ;
 printf("\n\nTotal Turn Around Time: %f",totaltat) ;
 printf("\nTotal Wait Time: %f",totalwt) ;
 printf("\n\nAverage Turn Around Time: %f",avgtat) ;
 printf("\nAverage Wait Time: %f\n",avgwt) ;
}

main()
{
 clrscr();
 getProcess();
 schduleProcess();
 display();
 getch();
 return 0;
}
```