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

#define MAX 10

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

process p[MAX] ;
int currenttime,processcount;
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) ;

 srand(time(NULL));
 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) ;
     do{
     p[i].bt=rand()%5;
     }while(p[i].bt==0);

     //printf("Enter the Arrival time: ") ;
     //scanf("%d",&p[i].at) ;
```

```
     p[i].at=rand()%10;

     p[i].tbt = p[i].bt ;
    }
 sort() ;
}

int getprocess()
{
 int i , min = 999 , p1=-1 ;
 for(i=0;i<processcount;i++)
    {
     if(p[i].at <= currenttime && p[i].tbt!=0)
       {
        if(p[i].tbt < min)
          {
           min = p[i].tbt ;
           p1 = i ;
          }
        }
    }
 return p1 ;
}


void scheduleprocess()
{
 int i,count=0;
 char currentprocess[10] , prevprocess[10] = "NULL" ;
 printf("\n\n  GanttChart:\n") ;
 printf("_____________________________________________________\n") ;
 while(1)
      {
        i = getprocess() ;
        if(i==-1)
        {
             strcpy(currentprocess,"idle") ;
             if(strcmp(currentprocess,prevprocess) !=0 )
               {
                printf("%d|%d  %s  ",currenttime , currenttime , currentprocess) ;
               }
             strcpy(prevprocess , currentprocess) ;
             currenttime++;
        }
        else
        {
            p[i].tbt-- ;
            strcpy(currentprocess,p[i].name) ;
            if(strcmp(currentprocess,prevprocess) !=0 )
              {
               printf("%d|%d  %s  ",currenttime , currenttime , currentprocess) ;
              }
            currenttime++ ;
            if(p[i].tbt==0)
              {
               p[i].ct = currenttime ;
               p[i].tat = p[i].ct - p[i].at ;//finishtime-arrival time
```

```
                p[i].wt = p[i].tat-p[i].bt ; //total existence-working time
                count++ ;
                totaltat += p[i].tat ;
                totalwt+=p[i].wt ;
               }
             strcpy(prevprocess , currentprocess) ;
             if(count==processcount)
               break ;
         }
     }
 printf("%d|",currenttime) ;
 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();
 readprocess();
 scheduleprocess();
 display();
 getch();
 return 0;
}
```