***ASSIGNMENT 1***

***SETA***

***Q1]***

#include<stdio.h>

#include<sys/types.h>

void ChildProcess();

void ParentProcess();

int main()

{

      pid\_t pid;

      pid =fork();

      if(pid==0)

      {

            printf("The process ID is %d\n",pid);

            ChildProcess();

      }

      else

      {

            printf("The process ID is %d\n",pid);

            ParentProcess();

      }

      return 0;

}

void ChildProcess()

{

      printf("I am child process\n");

}

void ParentProcess()

{

      printf("I am parent process\n");

}

***OUTPUT***

The process ID is 2405

I am parent process

The process ID is 0

I am child proces**s**

***Q2]***

#include<stdio.h>

main()

{

      int pid,retnice;

      printf("Press DEL to stop process\n");

      pid=fork();

      for(;;)

      {

            if(pid==0)

            {

                  retnice=nice(-5);

                  printf("Child gets higher CPU priority%d\n",retnice);

                  sleep(10);

            }

            else

            {

                  retnice=nice(4);

                  printf("Parent gets lower CPU priority%d\n",retnice);

                  sleep(10);

            }

      }

}

***OUTPUT***

Press DEL to stop process

Parent gets lower CPU priority4

Child gets higher CPU priority-1

^[[3~Parent gets lower CPU priority8

Child gets higher CPU priority-1

Child gets higher CPU priority-1

Parent gets lower CPU priority12

Child gets higher CPU priority-1

Parent gets lower CPU priority16

Child gets higher CPU priority-1

Parent gets lower CPU priority19

Child gets higher CPU priority-1

Parent gets lower CPU priority19

Parent gets lower CPU priority19

Child gets higher CPU priority-1

Parent gets lower CPU priority19

Child gets higher CPU priority-1

***Set B***

**Q2]**

**#include <stdio.h>**

**#include <sys/types.h>**

**#include <unistd.h>**

int main()

{

    // fork() Create a child process

    int pid = fork();

    if (pid > 0) {

        //getpid() returns process id

        // while getppid() will return parent process id

        printf("Parent process\n");

        printf("ID : %d\n\n", getpid());

    }

    else if (pid == 0) {

        printf("Child process\n");

        // getpid() will return process id of child process

        printf("ID: %d\n", getpid());

        // getppid() will return parent process id of child process

        printf("Parent -ID: %d\n\n", getppid());

        sleep(10);

        // At this time parent process has finished.

        // So if u will check parent process id

        // it will show different process id

        printf("\nChild process \n");

        printf("ID: %d\n", getpid());

        printf("Parent -ID: %d\n", getppid());

    }

    else {

        printf("Failed to create child process");

    }

    return 0;

**}**

***OUTPUT***

Parent process

ID : 3335

Child process

ID: 3336

Parent -ID: 3335  
**(set B Q1]remaining..)  
 *ASSIGNMENT 2***

***SET A***

**Q1]**

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

#include<fcntl.h>

#include<string.h>

#include<stdlib.h>

int main()

{

      char cmd[80],tok1[10],tok2[10],tok3[10],tok4[10];

      int n;

      while(1)

      {

            printf("\nMyShell$");

            fgets(cmd,80,stdin);

            n=sscanf(cmd,"%s%s%s%s",tok1,tok2,tok3,tok4);

            if(strcmp(tok1,"exit")==0)

                  exit(0);

            switch(n)

            {

                  case 1:

                        if(fork()==0)

                              execlp(tok1,tok2,tok3,tok4);

                        wait(0);

                        break;

                  case 2:

                        if(fork()==0)

                              execlp(tok1,tok1,tok2,NULL);

                        wait(0);

                        break;

                  case 3:

                        if(strcmp(tok1,"count")==0)

                              count(tok2,tok3);

                        else

                        {

                                    execlp(tok1,tok1,tok3,tok3,NULL);

                              wait(0);

                        break;

                        }

            }

      }

}

void count()

{

      int lc,cc,wc,fp;

      char ch;

        char tok2[10],tok3[10];

      lc=0;

      wc=0;

      cc=0;

      fp=open(tok3,O\_RDONLY);

      if(fp!=-1)

      {

            printf("\nFile %s not opened",tok3);

      }

      else

      {

            while(read(fp,&ch,1))

            {

                  if(ch!=' ' && ch!='\t' && ch!='\n')

                  {

                        cc++;

                  }

                  else if(ch=' ' || ch=='\t')

                  {

                        wc++;

                  }

                  else

                  {

                        wc++;

                        lc++;

                  }

            }

      }

      if(strcmp(tok2,"w")==0)

            printf("\n word count is:%d",wc);

      if(strcmp(tok2,"l")==0)

            printf("\n line count is:%d",lc);

      if(strcmp(tok2,"c")==0)

            printf("\nChar count is:%d",cc);

      close(fp);

}

***OUTPUT***

MyShell$count l shell.c

MyShell$count c cc.txt

**(Output not valid)**

**SET B**

**Q1]**

***1.List***

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <dirent.h>

void separate();

void list();

char cmd[80], token1[20],token2[20],token3[20],token4[20],ch;

int n,fp,wc,cc,lc;

pid\_t pid;

DIR\* dp;

struct dirent \*de;

void separate(){

strcpy(token1,"\0");

strcpy(token2,"\0");

strcpy(token3,"\0");

strcpy(token4,"\0");

n=sscanf(cmd,"%s%s%s%s", token1,token2,token3,token4);

}

int main(){

while(1){

printf("\n Myshell $");

fgets(cmd,80,stdin);

separate();

if (strcmp(token1,"exit")==0)

exit (0);

pid=fork();

if (pid ==-1)

printf("child process is not created");

else if(pid ==0){

if (strcmp(token1,"list")==0)

list();

else{

switch (n){

case 1:

if(fork()==0)

execlp(token1,token1,NULL);

wait(0);

break;

case 2:

if(fork()==0)

execlp(token1,token1,token2,NULL);

wait(0);

break;

case 3:

if(strcmp(token1,"list")==0)

list(token2,token3);

else

if(fork()==0)

execlp(token1,token1,token2,token3,NULL);

wait(0);

break;

case 4:

if(fork()==0)

execlp(token1,token1,token2,token3,token4,NULL);

wait(0);

break;

default:

printf("\nInvalid");

}

}

}

else

wait(NULL);

}

return 0;

}

void list(){

int cnt=0;

dp = opendir(token3);

if(dp==NULL){

printf("\n%s directory not found", token3);

}

else{

if(strcmp(token2 ,"f")==0){

while(1){

de = readdir(dp);

if (de==NULL)

break;

printf("\nfile name:%s",de->d\_name);

}

}

if (strcmp(token2,"n")==0){

while(1){

de=readdir(dp);

if(de==NULL){

break;

}

cnt++;

}

printf("\n total files :%d",cnt);

}

if (strcmp(token2,"i")==0){

while(1){

de=readdir(dp);

if(de==NULL){

break;

}

printf("\nfile name: %s",de->d\_name);

printf("\t inode no: %ld",de->d\_ino);

}

}

}

}

***OUTPUT***

 Myshell $list l new

 Myshell $list f new

file name:..

file name:.

file name:file.txt

file name:file1.txt

 Myshell $

1. ***search***

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <dirent.h>

void separate();

void search();

char cmd[80], token1[20],token2[20],token3[20],token4[20],ch;

int n,fp,wc,cc,lc;

pid\_t pid;

void separate(){

strcpy(token1,"\0");

strcpy(token2,"\0");

strcpy(token3,"\0");

strcpy(token4,"\0");

n=sscanf(cmd,"%s%s%s%s", token1,token2,token3,token4);

}

int main(){

while(1){

printf("\n Myshell $");

fgets(cmd,80,stdin);

separate();

if (strcmp(token1,"exit")==0)

exit (0);

pid=fork();

if (pid ==-1)

printf("child process is not created");

else if(pid ==0){

if (strcmp(token1,"search")==0)

search();

else{

switch (n){

case 1:

if(fork()==0)

execlp(token1,token1,NULL);

wait(0);

break;

case 2:

if(fork()==0)

execlp(token1,token1,token2,NULL);

wait(0);

break;

case 3:

if(strcmp(token1,"list")==0)

search(token2,token3);

else

if(fork()==0)

execlp(token1,token1,token2,token3,NULL);

wait(0);

break;

case 4:

if(fork()==0)

execlp(token1,token1,token2,token3,token4,NULL);

wait(0);

break;

default:

printf("\nInvalid");

}

}

}

else

wait(NULL);

}

return 0;

}

void search(){

char s[80];

int i=0,cnt=0,lc=1;

fp=open(token3, O\_RDONLY);

if(fp==-1){

printf("\nfile not found");

return;

}

while (read(fp,&ch,1)){

if(ch=='\n'){

lc++;

}

if(ch=='\n' || ch =='\t' || ch ==' '){

s[i]='\0';

i=0;

if(strstr(s,token4)){

if (strcmp(token2, "f")==0){

printf("pattern\%s\ found in \%s\ at line no:%d\n",token4,s,lc);

break;

}

if (strcmp(token2,"c")==0){

cnt++;

}

if (strcmp(token2,"a")==0){

printf("pattern\%s\ found in \%s\ at line no:%d\n",token4,s,lc);

}

}

}

else{

s[i]=ch;

i++;

}

}

if (strcmp(token2,"c")==0){

printf("total occurences of given pattern=%d",cnt);

}

}

***OUTPUT***

 Myshell $search a file.txt

pattern found in this at line no:1

pattern found in is at line no:1

pattern found in the at line no:1

pattern found in first at line no:1

pattern found in line at line no:1

pattern found in of at line no:1

pattern found in code at line no:1

pattern found in  at line no:2

pattern found in and at line no:2

pattern found in i at line no:2

pattern found in am at line no:2

pattern found in working at line no:2

pattern found in on at line no:2

pattern found in the at line no:2

pattern found in linux at line no:2

pattern found in operating at line no:2

pattern found in system at line no:3

 Myshell $

1. ***Typeline***

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <dirent.h>

void separate();

void typeline();

char cmd[80], token1[20],token2[20],token3[20],token4[20],ch;

int n,fp,wc,cc,lc,m;

pid\_t pid;

void separate(){

strcpy(token1,"\0");

strcpy(token2,"\0");

strcpy(token3,"\0");

strcpy(token4,"\0");

n=sscanf(cmd,"%s%s%s%s", token1,token2,token3,token4);

}

int main(){

while(1){

printf("\n Myshell $");

fgets(cmd,80,stdin);

separate();

if (strcmp(token1,"exit")==0)

exit (0);

pid=fork();

if (pid ==-1)

printf("child process is not created");

else if(pid ==0){

if (strcmp(token1,"search")==0)

typeline();

else{

switch (n){

case 1:

if(fork()==0)

execlp(token1,token1,NULL);

wait(0);

break;

case 2:

if(fork()==0)

execlp(token1,token1,token2,NULL);

wait(0);

break;

case 3:

if(strcmp(token1,"typeline")==0)

typeline(token2,token3);

else

if(fork()==0)

execlp(token1,token1,token2,token3,NULL);

wait(0);

break;

case 4:

if(fork()==0)

execlp(token1,token1,token2,token3,token4,NULL);

wait(0);

break;

default:

printf("\nInvalid");

}

}

}

else

wait(NULL);

}

return 0;

}

void typeline(){

lc=0;

fp=open(token3,O\_RDONLY);

while (read(fp,&ch,1)){

if (ch=='\n'){

lc++;

}

lc=lc+1;

lseek(fp, 0, SEEK\_SET);

if(strcmp(token2,"a")==0){

while(read(fp,&ch,1)){

printf("%c",ch);

}

}

else{

n=atoi(token2);

if (n>0){

while(read(fp,&ch,1)){

printf("%c",ch);

if (ch=='\n'){

n--;

}

if(n==0){

break;}

}

}

else{

n=-n;

m=lc-n;

while(read(fp,&ch,1)){

if(ch=='\n'){

m--;

}

if(m==0){

break;

}

}

while(read(fp,&ch,1)){

printf("%c",ch);

}

}

}

close(fp);

}

}

***OUTPUT***

 Myshell $typeline a file.txt

this is the first line of code

and i am working on the linux operating system

***ASSIGNMENT 3***

***SET A***

***Q1]***

#include<stdio.h>

#include<string.h>

struct job

{

      char name[20];

      int at,bt,ct,tat,wt,st,tbt;

}job[10];

int n,i,j;

float avg\_tat=0;

float avg\_wt=0;

void take\_input()

{

      printf("\n Enter the no.of process:");

      scanf("%d",&n);

      for(i=0;i<n;i++)

      {

            printf("Enter the arrival time of the process%d",i);

            scanf("%d",&job[i].at);

      }

      for(i=0;i<n;i++)

      {

            printf("Enter the burst time of the process %d",i);

            scanf("%d",&job[i].bt);

            job[i].tbt=job[i].bt;

      }

      for(i=0;i<n;i++)

      {

            printf("Enter the name of process %d",i);

            scanf("%s",&job[i].name);

      }

}

void sort()

{

      struct job temp;

      for(i=0;i<n;i++)

      {

            for(j=i+1;j<n;j++)

            {

                  if(job[i].at>job[j].at)

                  {

                        temp=job[i];

                        job[i]=job[j];

                        job[j]=temp;

                  }

            }

      }

}

void process()

{

      int time=job[0].at;

      for(j=0;j<n;j++)

      {

            job[j].st=time;

            printf("| %d %s",job[j].st,job[j].name);

            time=time+job[j].tbt;

            job[j].ct=time;

            job[j].tat=time-job[j].at;

            job[j].wt=job[j].tat-job[j].tbt;

            printf("%d |",time);

      }

}

void print\_output()

{

      printf("\n \n");

      printf("\n--------------------------------------------------------------");

      printf("\nPname                   At                      Bt                    Tat                      Wt");

      printf("\n-------------------------------------------------------------");

      for(i=0;i<n;i++)

      {

            printf("\n%s        %8d           %5d                  %4d                 %3d",job[i].name,job[i].at,job[i].bt,job[i].tat,job[i].wt);

            avg\_tat=avg\_tat+(float)(job[i].tat);

            avg\_wt=(float)avg\_wt+(float)(job[i].wt);

      }

      printf("\n---------------------------------------------------");

      printf("\n the avg of the turn around time is %f",avg\_tat/n);

      printf("\n the avg of the waiting time is %f",avg\_wt/n);

}

int main()

{

      int i;

      take\_input();

      sort();

      process();

      print\_output();

      printf("\n \n");

      for(i=0;i<n;i++)

      {

            job[i].tbt=job[i].bt=rand()%10+1;

            job[i].at=job[i].ct+2;

      }

      process();

      print\_output();

}

***OUTPUT***

Enter the no.of process:5

Enter the arrival time of the process01

Enter the arrival time of the process14

Enter the arrival time of the process25

Enter the arrival time of the process36

Enter the arrival time of the process48

Enter the burst time of the process 04

Enter the burst time of the process 16

Enter the burst time of the process 27

Enter the burst time of the process 38

Enter the burst time of the process 42

Enter the name of process 01

Enter the name of process 12

Enter the name of process 23

Enter the name of process 34

Enter the name of process 45

| 1 15 || 5 211 || 11 318 || 18 426 || 26 528 |

--------------------------------------------------------------

Pname                   At                      Bt                    Tat                      Wt

-------------------------------------------------------------

1               1               4                     4                   0

2               4               6                     7                   1

3               5               7                    13                   6

4               6               8                    20                  12

5               8               2                    20                  18

---------------------------------------------------

 the avg of the turn around time is 12.800000

 the avg of the waiting time is 7.400000

| 7 111 || 11 218 || 18 326 || 26 432 || 32 536 |

--------------------------------------------------------------

Pname                   At                      Bt                    Tat                      Wt

-------------------------------------------------------------

1               7               4                     4                   0

2              13               7                     5                  -2

3              20               8                     6                  -2

4              28               6                     4                  -2

5              30               4                     6                   2

---------------------------------------------------

 the avg of the turn around time is 17.799999

 the avg of the waiting time is 6.600000

***Q2]***

#include<stdio.h>

#include<string.h>

struct job

{

      char name[20];

      int at,bt,ct,tat,wt,st,tbt;

}job[10];

int n,i,j;

float avg\_tat=0;

float avg\_wt=0;

void take\_input()

{

      printf("\n Enter the no.of process:");

      scanf("%d",&n);

      for(i=0;i<n;i++)

      {

            printf("Enter the arrival time of the process%d",i);

            scanf("%d",&job[i].at);

      }

      for(i=0;i<n;i++)

      {

            printf("Enter the burst time of the process %d",i);

            scanf("%d",&job[i].bt);

            job[i].tbt=job[i].bt;

      }

      for(i=0;i<n;i++)

      {

            printf("Enter the name of process %d",i);

            scanf("%s",&job[i].name);

      }

}

void sort()

{

      struct job temp;

      for(i=0;i<n;i++)

      {

            for(j=i+1;j<n;j++)

            {

                  if(job[i].bt>job[j].bt)

                  {

                        temp=job[i];

                        job[i]=job[j];

                        job[j]=temp;

                  }

            }

      }

}

void process()

{

      int time=job[0].at;

      for(j=0;j<n;j++)

      {

            job[j].st=time;

            printf("| %d %s",job[j].st,job[j].name);

            time=time+job[j].tbt;

            job[j].ct=time;

            job[j].tat=time-job[j].at;

            job[j].wt=job[j].tat-job[j].tbt;

            printf("%d |",time);

      }

}

void print\_output()

{

      printf("\n \n");

      printf("\n--------------------------------------------------------------");

      printf("\nPname                   At                      Bt                    Tat                      Wt");

      printf("\n-------------------------------------------------------------");

      for(i=0;i<n;i++)

      {

            printf("\n%s        %8d           %5d                  %4d                 %3d",job[i].name,job[i].at,job[i].bt,job[i].tat,job[i].wt);

            avg\_tat=avg\_tat+(float)(job[i].tat);

            avg\_wt=(float)avg\_wt+(float)(job[i].wt);

      }

      printf("\n---------------------------------------------------");

      printf("\n the avg of the turn around time is %f",avg\_tat/n);

      printf("\n the avg of the waiting time is %f",avg\_wt/n);

}

int main()

{

      int i;

      take\_input();

      print\_output();

      sort();

      process();

      print\_output();

      printf("\n \n");

      for(i=0;i<n;i++)

      {

            job[i].tbt=job[i].bt=rand()%10+1;

            job[i].at=job[i].ct+2;

            }

      process();

      print\_output();

}

***OUTPUT***

 Enter the no.of process:4

Enter the arrival time of the process02

Enter the arrival time of the process13

Enter the arrival time of the process21

Enter the arrival time of the process30

Enter the burst time of the process 02

Enter the burst time of the process 13

Enter the burst time of the process 24

Enter the burst time of the process 38

Enter the name of process 00

Enter the name of process 11

Enter the name of process 22

Enter the name of process 33

--------------------------------------------------------------

Pname                   At                      Bt                    Tat                      Wt

-------------------------------------------------------------

0               2               2                     0                   0

1               3               3                     0                   0

2               1               4                     0                   0

3               0               8                     0                   0

---------------------------------------------------

 the avg of the turn around time is 0.000000

 the avg of the waiting time is 0.000000| 2 04 || 4 17 || 7 211 || 11 319 |

--------------------------------------------------------------

Pname                   At                      Bt                    Tat                      Wt

-------------------------------------------------------------

0               2               2                     2                   0

1               3               3                     4                   1

2               1               4                    10                   6

3               0               8                    19                  11

---------------------------------------------------

 the avg of the turn around time is 8.750000

 the avg of the waiting time is 4.500000

| 6 010 || 10 117 || 17 225 || 25 331 |

--------------------------------------------------------------

Pname                   At                      Bt                    Tat                      Wt

-------------------------------------------------------------

0               6               4                     4                   0

1               9               7                     8                   1

2              13               8                    12                   4

3              21               6                    10                   4

---------------------------------------------------

 the avg of the turn around time is 17.250000

 the avg of the waiting time is 6.75000

**(OUTPUT INVALID)**

***SETB***

***Q1]***

#include<stdio.h>

#include<stdlib.h>

struct Process

{

      int first\_bt,next\_bt,at,ft,tat,wt,temp\_bt;

}

P[100];

struct Schedule

{

      int pid,endtime;

}

sch[100];

int sch\_cnt;

int ct;

int np;

int i;

void accept()

{

      printf("Enter the number of process:");

      scanf("%d",&np);

      for(i=0;i<np;i++)

      {

            printf("Enter the arrival time:");

            scanf("%d",&P[i].at);

            printf("Enter the first burst time:");

            scanf("%d",&P[i].first\_bt);

            P[i].next\_bt=rand()%10+1;

            P[i].temp\_bt=P[i].first\_bt;

      }

}

int getProcessBySJF()

{

      int minp=-1;

      int minat=32768;

      for(i=0;i<np;i++)

      {

            if(P[i].at<=ct && P[i].temp\_bt>0 && P[i].temp\_bt<minat)

            {

                  minp=i;

                  minat=P[i].temp\_bt;

            }

      }

      return minp;

}

void showGanttChart()

{

      for(i=0;i<sch\_cnt\*5;i++)

            printf("--");

            printf("\n|");

      for(i=0;i<sch\_cnt;i++)

            printf("P%d |",sch[i].pid);

            printf("\n");

      for(i=0;i<sch\_cnt\*5;i++)

            printf("--");

            printf("\n0");

      for(i=0;i<sch\_cnt;i++)

            printf("%5d",sch[i].endtime);

}

void output()

{

      float avgtat=0,avgwt=0;

      printf("\n");

      printf("\nPID\tFBT\tRBT\tAT\tFT\tTAT\tWT\n");

      for(i=0;i<np;i++)

      {

            printf("\nP%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",i,P[i].first\_bt,P[i].next\_bt,P[i].at,P[i].ft,P[i].tat,P[i].wt);

            avgtat+=P[i].tat;

            avgwt+=P[i].wt;

      }

      avgtat=avgtat/np;

      avgwt=avgwt/np;

      printf("Average turn around time:%f",avgtat);

      printf("\nAverage waiting time:%f\n",avgwt);

}

int main()

{

      accept();

      while(1)

      {

            int x=getProcessBySJF();

            if(x==-1)

                  break;

            sch[sch\_cnt].pid=x;

            ct+=P[x].temp\_bt;

            P[x].temp\_bt=0;

            P[x].ft=ct;

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

            P[x].tat=P[x].ft-P[x].at;

            P[x].wt=P[x].tat-P[x].first\_bt;

      }

      showGanttChart();

      output();

      while(1)

      {

            int x=getProcessBySJF();

            if(x==-1)

                  break;

            sch[sch\_cnt].pid=x;

            ct+=P[x].temp\_bt;

            P[x].temp\_bt=0;

            P[x].ft=ct;

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

            P[x].tat=P[x].ft-P[x].at;

            P[x].wt=P[x].tat-P[x].first\_bt-P[x].next\_bt-2;

      }

      showGanttChart();

      output();

}

***OUTPUT***

Enter the number of process:4

Enter the arrival time:2

Enter the first burst time:2

Enter the arrival time:

3

Enter the first burst time:3

Enter the arrival time:1

Enter the first burst time:4

Enter the arrival time:0

Enter the first burst time:8

----------------------------------------

|P3 |P0 |P1 |P2 |

----------------------------------------

0    8   10   13   17

PID   FBT   RBT   AT    FT    TAT   WT

P0    2     4     2     10    8     6     0     0

P1    3     7     3     13    10    7     0     1086324736

P2    4     8     1     17    16    12    0     1095761920

P3    8     6     0     8     8     0     0     1103626240

Average turn around time:10.500000

Average waiting time:6.250000

----------------------------------------

|P3 |P0 |P1 |P2 |

----------------------------------------

0    8   10   13   17

PID   FBT   RBT   AT    FT    TAT   WT

P0    2     4     2     10    8     6     0     0

P1    3     7     3     13    10    7     0     1086324736

P2    4     8     1     17    16    12    0     1095761920

P3    8     6     0     8     8     0     0     1103626240

Average turn around time:10.500000

Average waiting time:6.250000

***Q2]***

#include<stdio.h>

#include<stdlib.h>

struct Process

{

      int first\_bt,next\_bt,at,ft,tat,wt,temp\_bt,prio;

}

P[100];

struct Schedule

{

      int pid,endtime;

}

sch[100];

int sch\_cnt;

int ct;

int np;

int i;

void accept()

{

      printf("Enter the number of process:");

      scanf("%d",&np);

      for(i=0;i<np;i++)

      {

            printf("Enter the arrival time:");

            scanf("%d",&P[i].at);

            printf("Enter the first burst time:");

            scanf("%d",&P[i].first\_bt);

            P[i].next\_bt=rand()%10+1;

            P[i].temp\_bt=P[i].first\_bt;

            printf("Enter the process of priority:");

            scanf("%d",&P[i].prio);

      }

}

int getProcessByPriority()

{

      int minp=-1;

      int minat=32768;

      for(i=0;i<np;i++)

      {

            if(P[i].at<=ct && P[i].temp\_bt>0 && P[i].prio<minat)

            {

                  minp=i;

                  minat=P[i].prio;

            }

      }

      return minp;

}

void showGanttChart()

{

      for(i=0;i<sch\_cnt\*5;i++)

            printf("--");

            printf("\n|");

      for(i=0;i<sch\_cnt;i++)

            printf("P%d |",sch[i].pid);

            printf("\n");

      for(i=0;i<sch\_cnt\*5;i++)

            printf("--");

            printf("\n0");

      for(i=0;i<sch\_cnt;i++)

            printf("%5d",sch[i].endtime);

}

void output()

{

      float avgtat=0,avgwt=0;

      printf("\n");

      printf("\nPID\tFBT\tRBT\tAT\tFT\tTAT\tWT\n");

      for(i=0;i<np;i++)

      {

            printf("\nP%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",i,P[i].first\_bt,P[i].next\_bt,P[i].at,P[i].ft,P[i].tat,P[i].wt,P[i].prio);

            avgtat+=P[i].tat;

            avgwt+=P[i].wt;

      }

      avgtat=avgtat/np;

      avgwt=avgwt/np;

      printf("Average turn around time:%f",avgtat);

      printf("\nAverage waiting time:%f\n",avgwt);

}

int main()

{

      accept();

      while(1)

      {

            int x=getProcessByPriority();

            if(x==-1)

                  break;

            sch[sch\_cnt].pid=x;

            ct+=P[x].temp\_bt;

            P[x].temp\_bt=0;

            P[x].ft=ct;

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

            P[x].tat=P[x].ft-P[x].at;

            P[x].wt=P[x].tat-P[x].first\_bt;

      }

      showGanttChart();

      output();

      while(1)

      {

            int x=getProcessByPriority();

            if(x==-1)

                  break;

            sch[sch\_cnt].pid=x;

            ct+=P[x].temp\_bt;

            P[x].temp\_bt=0;

            P[x].ft=ct;

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

            P[x].tat=P[x].ft-P[x].at;

            P[x].wt=P[x].tat-P[x].first\_bt-P[x].next\_bt-2;

      }

      showGanttChart();

      output();

}

***OUTPUT***

Enter the number of process:4

Enter the arrival time:2

Enter the first burst time:2

Enter the process of priority:0

Enter the arrival time:1

Enter the first burst time:3

Enter the process of priority:1

Enter the arrival time:0

Enter the first burst time:4

Enter the process of priority:2

Enter the arrival time:4

Enter the first burst time:5

Enter the process of priority:3

----------------------------------------

|P2 |P0 |P1 |P3 |

----------------------------------------

0    4    6    9   14

PID   FBT   RBT   AT    FT    TAT   WT

P0    2     4     2     6     4     2     0     624371122   0

P1    3     7     1     9     8     5     1     624371122   0

P2    4     8     0     4     4     0     2     624371122   0

P3    5     6     4     14    10    5     3     624371122   0

Average turn around time:6.500000

Average waiting time:3.000000

----------------------------------------

|P2 |P0 |P1 |P3 |

----------------------------------------

0    4    6    9   14

PID   FBT   RBT   AT    FT    TAT   WT

P0    2     4     2     6     4     2     0     624371122   0

P1    3     7     1     9     8     5     1     624371122   0

P2    4     8     0     4     4     0     2     624371122   0

P3    5     6     4     14    10    5     3     624371122   0

Average turn around time:6.500000

Average waiting time:3.000000

***SET C***

***Q1]***

#include<stdio.h>

#include<stdlib.h>

struct Process

{

      int first\_bt, next\_bt, at, ft, tat, wt, temp\_bt, priority;

}P[100];

struct Schedule

{

      int pid, endtime;

}sch[100];

int sch\_cnt;    //Schedule Counter

int ct;        //Current time

int np;       //Number of processes

int i;

void accept()

{

      printf("Enter the number of processes:");

      scanf("%d",&np);

      for(i=0;i<np;i++)

      {

            printf("Enter the arrival time:");

            scanf("%d",&P[i].at);

            printf("Enter the first burst time:");

            scanf("%d",&P[i].first\_bt);

        printf("Enter the priority:");

        scanf("%d",&P[i].priority);

        P[i].next\_bt=rand()%5+1;

        P[i].temp\_bt=P[i].first\_bt;

      }

}

int getProcessByPriority()

{

      int minp=-1;

      int minat=32768;

      for(i=0;i<np;i++)

      {

            if(P[i].at<=ct && P[i].temp\_bt>0 && P[i].priority<minat)

            {

                  minp=i;

                  minat=P[i].priority;

            }

      }

    return minp;

}

void showGanttChart()

{

      for(i=0;i<sch\_cnt\*5;i++)

      printf("-");

      printf("\n|");

    for(i=0;i<sch\_cnt;i++)

      printf(" P%d |",sch[i].pid);

      printf("\n");

    for(i=0;i<sch\_cnt\*5;i++)

      printf("-");

      printf("\n0");

      for(i=0;i<sch\_cnt;i++)

      printf("%5d",sch[i].endtime);

}

void showPT()

{

      float avgtat,avgwt;

    printf("\n");

      printf("\nPID\tFBT\tNBT\tAT\tFT\tTAT\tWT\tPriority\n");

      for(i=0;i<np;i++)

      {

            printf("\nP%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",i,P[i].first\_bt,P[i].next\_bt,P[i].at,P[i].ft,P[i].tat,P[i].wt,P[i].priority);

            avgtat+=P[i].tat;

            avgwt+=P[i].wt;

      }

      avgtat=avgtat/np;

      avgwt=avgwt/np;

      printf("Average turn arount time:%f",avgtat);

      printf("\nAverage waiting time:%f\n",avgwt);

}

void main()

{

      accept();

      while(1)

      {

            int x=getProcessByPriority();

            if(x==-1)              //if no process, will return -1

                  break;

            sch[sch\_cnt].pid=x;

            ct++; // In case of Preemptive

        // ct = ct + P[x].temp\_bt;   In cast of NonPreemptive

            P[x].temp\_bt--; // In case of Preemptive

        // P[x].temp\_bt = 0;  In case of NonPreemptive

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

      }

      for(i=0;i<np;i++)

      {

            P[i].temp\_bt=P[i].next\_bt;

      }

      while(1)

      {

            int x=getProcessByPriority();

            if(x==-1)

                  break;

            sch[sch\_cnt].pid=x;

            ct++;

            P[x].temp\_bt--;

            sch[sch\_cnt].endtime=ct;

            sch\_cnt++;

        if(P[x].temp\_bt==0)

        {

            P[x].ft=ct;

                P[x].tat=P[x].ft-P[x].at;

                P[x].wt=P[x].tat-P[x].first\_bt-P[x].next\_bt-2;

        }

      }

      showGanttChart();

      showPT();

}

***OUTPUT***

Enter the number of processes:4

Enter the arrival time:2

Enter the first burst time:2

Enter the priority:0

Enter the arrival time:1

Enter the first burst time:3

Enter the priority:1

Enter the arrival time:0

Enter the first burst time:4

Enter the priority:2

Enter the arrival time:4

Enter the first burst time:5

Enter the priority:3

------------------------------------------------------------------------------------------------------------------------

| P2 | P1 | P0 | P0 | P1 | P1 | P2 | P2 | P2 | P3 | P3 | P3 | P3 | P3 | P0 | P0 | P0 | P0 | P1 | P1 | P2 | P2 | P2 | P3 |

------------------------------------------------------------------------------------------------------------------------

0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24

PID   FBT   NBT   AT    FT    TAT   WT    Priority

P0    2     4     2     18    16    8     0

P1    3     2     1     20    19    12    1

P2    4     3     0     23    23    14    2

P3    5     1     4     24    20    12    3

Average turn arount time:19.500000

Average waiting time:11.500000

***Q2]***

#include<stdio.h>

#include<string.h>

struct process

{

      char name[20];

      int at,bt,ft,tat,wt,st,temp\_bt;

}p[10];

int np,i,j,tq;

float avg\_tat=0;

float avg\_wt=0;

// to accept the info about pes

void take\_input()

{

      printf("Enter the no of ps : ");

      scanf("%d",&np);

      printf("Enter the time Quantum: ");

      scanf("%d",&tq);

      for(i=0;i<np;i++)

      {

            printf("Enter the name of the p: ");

            scanf("%s",&p[i].name);

            printf("Enter the arrival time of the p : ");

            scanf("%d",&p[i].at);

            printf("Enter the burst time of the p: ");

            scanf("%d",&p[i].bt);

            p[i].temp\_bt=p[i].bt;

            printf("\n\n");

      }

}

// to sort the pes by arriaval time

void sort()

{

      struct process temp;

      for(i=0;i<np;i++)

      {

            for(j=i+1;j<np;j++)

            {

                  if(p[i].at>p[j].at)

                  {

                        temp=p[i];

                        p[i]=p[j];

                        p[j]=temp;

                  }

            }

      }

}

// to calculate the tat n wt

void getprocessbyRR()

{

      int x=0,cnt=0,time=p[0].at;

      printf("\n\*\*\*\*\*\*\*\*\*\*\*Gantt chart\*\*\*\*\*\*\*\*\*\n");

      while(1)

      {

            if(p[x].temp\_bt!=0)

            {

                  printf("|%d %s ",time,p[x].name);

                  if(p[x].temp\_bt>=tq)

                  {

                        p[x].temp\_bt=p[x].temp\_bt-tq;

                        time=time+tq;

                  }

                  else

                  {

                        time=time+p[x].temp\_bt;

                        p[x].temp\_bt=0;

                  }

                  printf("%d |",time);

                  if(p[x].temp\_bt==0)

                  {

                        p[x].ft=time;

                        p[x].tat=time-p[x].at;

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

                        cnt++;

                  }

            }

            x++;

            if(x==np)

            {

                  x=0;

            }

            if(cnt==np)

                  break;

      }

}

//to print the output table

void print\_output()

{

      printf("\n\n");

      printf("\n---------------------------------------");

      printf("\n pname\tat\temp\_bt\ttat\twt ");

      printf("\n---------------------------------------");

      for(i=0;i<np;i++)

      {

            printf("\n%s\t%d\t%d\t%d\t%d",p[i].name,p[i].at,p[i].bt,p[i].tat,p[i].wt);

            avg\_tat=avg\_tat+(float)(p[i].tat);

            avg\_wt=(float)avg\_wt+(float)(p[i].wt);

      }

      printf("\n-----------------------------------------");

      printf("\nThe avg of the turn around time is %f",avg\_tat/np);

      printf("\nThe avg of the waiting time is %f",avg\_wt/np);

}

void main()

{

      int i;

      take\_input();

      sort();

      getprocessbyRR();

      print\_output();

      for(i=0;i<np;i++)

      {

            p[i].temp\_bt=p[i].bt=rand()%10+1;

            p[i].at=p[i].bt+2;

      }

      getprocessbyRR();

      print\_output();

}

***OUTPUT***