OS ASSIGNMENT 6

NAME: Kaushal Oza

Roll no: 40

SRN: 201900754

DIV A

CPU Scheduling:

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

typedef struct process{
        char name[5];
        int bt;
        int at;
        int prt;
        int wt,ta;
        int flag;
}processes;


void b_sort(processes temp[],int n)
{
        processes t;
        int i,j;
        for(i=1;i<n;i++)
                for(j=0;j<n-i;j++){
```

```
                        if(temp[j].at > temp[j+1].at){
                                t = temp[j];
                                temp[j] = temp[j+1];
                                temp[j+1] = t;
                        }
                }
}

int accept(processes P[]){
        int i,n;
        printf("\n Enter total no. of processes : ");
        scanf("%d",&n);
        for(i=0;i<n;i++){
                printf("\n PROCESS [%d]",i+1);
                printf(" Enter process name : ");
                scanf("%s",&P[i].name);
                printf(" Enter burst time : ");
                scanf("%d",&P[i].bt);
                printf(" Enter arrival time : ");
                scanf("%d",&P[i].at);
                printf(" Enter priority : ");
                scanf("%d",&P[i].prt);
        }
        printf("\n PROC.\tB.T.\tA.T.\tPRIORITY");
        for(i=0;i<n;i++)
                printf("\n %s\t%d\t%d\t%d",P[i].name,P[i].bt,P[i].at,P[i].prt);
        return n;
}
```

```
void FCFS(processes P[],int n){
	processes temp[10];
	int sumw=0,sumt=0;
	int x = 0;
	float avgwt=0.0,avgta=0.0;
	int i,j;
	for(i=0;i<n;i++)
		temp[i]=P[i];

	b_sort(temp,n);

		printf("\n\n PROC.\tB.T.\tA.T.");
		for(i=0;i<n;i++)
			printf("\n %s\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

		sumw = temp[0].wt = 0;
		sumt = temp[0].ta = temp[0].bt - temp[0].at;

		for(i=1;i<n;i++){
			temp[i].wt = (temp[i-1].bt + temp[i-1].at + temp[i-1].wt) - temp[i].at;;
			temp[i].ta = (temp[i].wt + temp[i].bt);
			sumw+=temp[i].wt;
			sumt+=temp[i].ta;
		}
		avgwt = (float)sumw/n;
		avgta = (float)sumt/n;
		printf("\n\n PROC.\tB.T.\tA.T.\tW.T\tT.A.T");
		for(i=0;i<n;i++)
```

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


                printf("\n\n GANTT CHART\n ");
                for(i=0;i<n;i++)
                        printf("   %s   ",temp[i].name);
                printf("\n ");


                printf("0\t");
                for(i=1;i<=n;i++){
                        x+=temp[i-1].bt;
                        printf("%d      ",x);
                }
                printf("\n\n Average waiting time = %0.2f\n Average turn-around =
%0.2f.",avgwt,avgta);
}


void SJF_NP(processes P[],int n){
        processes temp[10];
        processes t;
        int sumw=0,sumt=0;
        int x = 0;
        float avgwt=0.0,avgta=0.0;
        int i,j;


        for(i=0;i<n;i++)
                temp[i]=P[i];


        b_sort(temp,n);
```

```
for(i=2;i<n;i++)
        for(j=1;j<n-i+1;j++){
                if(temp[j].bt > temp[j+1].bt){
                        t = temp[j];
                        temp[j] = temp[j+1];
                        temp[j+1] = t;
                }
        }

printf("\n\n PROC.\tB.T.\tA.T.");
        for(i=0;i<n;i++)
                printf("\n %s\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

        sumw = temp[0].wt = 0;
        sumt = temp[0].ta = temp[0].bt - temp[0].at;

        for(i=1;i<n;i++){
                temp[i].wt = (temp[i-1].bt + temp[i-1].at + temp[i-1].wt) - temp[i].at;;
                temp[i].ta = (temp[i].wt + temp[i].bt);
                sumw+=temp[i].wt;
                sumt+=temp[i].ta;
        }
        avgwt = (float)sumw/n;
        avgta = (float)sumt/n;
        printf("\n\n PROC.\tB.T.\tA.T.\tW.T\tT.A.T");
        for(i=0;i<n;i++)
                printf("\n
%s\t%d\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].wt,temp[i].ta);
```

```
		printf("\n\n GANTT CHART\n ");
		for(i=0;i<n;i++)
			printf("   %s   ",temp[i].name);
		printf("\n ");

		printf("0\t");
		for(i=1;i<=n;i++){
			x+=temp[i-1].bt;
			printf("%d      ",x);
		}
		printf("\n\n Average waiting time = %0.2f\n Average turn-around =
%0.2f.",avgwt,avgta);
}

void PRT_NP(processes P[],int n)
{
	processes temp[10];
	processes t;
	int sumw=0,sumt=0;
	float avgwt=0.0,avgta=0.0;
	int i,j;
	int x = 0;

	for(i=0;i<n;i++)
		temp[i]=P[i];

	b_sort(temp,n);
```

```
	for(i=2;i<n;i++)
		for(j=1;j<n-i+1;j++){
			if(temp[j].prt > temp[j+1].prt){
				t = temp[j];
				temp[j] = temp[j+1];
				temp[j+1] = t;
			}
		}

	printf("\n\n PROC.\tB.T.\tA.T.");
		for(i=0;i<n;i++)
			printf("\n %s\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

		sumw = temp[0].wt = 0;
		sumt = temp[0].ta = temp[0].bt - temp[0].at;

		for(i=1;i<n;i++){
			temp[i].wt = (temp[i-1].bt + temp[i-1].at + temp[i-1].wt) - temp[i].at;;
			temp[i].ta = (temp[i].wt + temp[i].bt);
			sumw+=temp[i].wt;
			sumt+=temp[i].ta;
		}
		avgwt = (float)sumw/n;
		avgta = (float)sumt/n;
		printf("\n\n PROC.\tB.T.\tA.T.\tW.T\tT.A.T");
		for(i=0;i<n;i++)
			printf("\n
%s\t%d\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].wt,temp[i].ta);
```

```
		printf("\n\n GANTT CHART\n ");
		for(i=0;i<n;i++)
			printf("   %s   ",temp[i].name);
		printf("\n ");

		printf("0\t");
		for(i=1;i<=n;i++){
			x+=temp[i-1].bt;
			printf("%d      ",x);
		}
		printf("\n\n Average waiting time = %0.2f\n Average turn-around =
%0.2f.",avgwt,avgta);
}

void RR(processes P[],int n)
{
	int pflag=0,t,tcurr=0,k,i,Q=0;
	int sumw=0,sumt=0;
	float avgwt=0.0,avgta=0.0;
	processes temp1[10],temp2[10];

	for(i=0;i<n;i++)
		temp1[i]=P[i];

	b_sort(temp1,n);

	for(i=0;i<n;i++)
		temp2[i]=temp1[i];
```

```
printf("\n Enter quantum time : ");
scanf("%d",&Q);


for(k=0;;k++){
        if(k>n-1)
                k=0;
        if(temp1[k].bt>0)
                printf("  %d  %s",tcurr,temp1[k].name);
        t=0;
        while(t<Q && temp1[k].bt > 0){
                t++;
                tcurr++;
                temp1[k].bt--;
        }
        if(temp1[k].bt <= 0 && temp1[k].flag != 1){
                temp1[k].wt = tcurr - temp2[k].bt - temp1[k].at;
                temp1[k].ta = tcurr - temp1[k].at;
                pflag++;
                temp1[k].flag = 1;
                sumw+=temp1[k].wt;
                sumt+=temp1[k].ta;
        }
        if(pflag == n)
                break;
}
printf("  %d",tcurr);
avgwt = (float)sumw/n;
avgta = (float)sumt/n;
printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);
```

```
}

void SJF_P(processes P[],int n){
	int i,t_total=0,tcurr,b[10],min_at,j,x,min_bt;
	int sumw=0,sumt=0;
	float avgwt=0.0,avgta=0.0;
	processes temp[10],t;

	for(i=0;i<n;i++){
		temp[i]=P[i];
		t_total+=P[i].bt;
	}

	b_sort(temp,n);

	for(i=0;i<n;i++)
		b[i] = temp[i].bt;

	i=j=0;
	printf("\n GANTT CHART\n\n %d %s",i,temp[i].name);
	for(tcurr=0;tcurr<t_total;tcurr++){

		if(b[i] > 0 && temp[i].at <= tcurr)
			b[i]--;

		if(i!=j)
			printf(" %d %s",tcurr,temp[i].name);

		if(b[i]<=0 && temp[i].flag != 1){
```

```
                temp[i].flag = 1;
                temp[i].wt = (tcurr+1) - temp[i].bt - temp[i].at;
                temp[i].ta = (tcurr+1) - temp[i].at;
                sumw+=temp[i].wt;
                sumt+=temp[i].ta;
            }
            j=i;    min_bt = 999;
            for(x=0;x<n;x++){

                if(temp[x].at <= (tcurr+1) && temp[x].flag != 1){

                    if(min_bt != b[x] && min_bt > b[x]){
                        min_bt = b[x];
                        i=x;
                    }
                }
            }

        }
        printf(" %d",tcurr);
        avgwt = (float)sumw/n; avgta = (float)sumt/n;
        printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);
}


void PRT_P(processes P[],int n){
        int i,t_total=0,tcurr,b[10],j,x,min_pr;
        int sumw=0,sumt=0;
```

```
float avgwt=0.0,avgta=0.0;
processes temp[10],t;

for(i=0;i<n;i++){
        temp[i]=P[i];
        t_total+=P[i].bt;
}

b_sort(temp,n);

for(i=0;i<n;i++)
        b[i] = temp[i].bt;

i=j=0;
printf("\n GANTT CHART\n\n %d %s",i,temp[i].name);
for(tcurr=0;tcurr<t_total;tcurr++)
{

        if(b[i] > 0 && temp[i].at <= tcurr)
                b[i]--;

        if(i!=j)
                printf(" %d %s",tcurr,temp[i].name);

        if(b[i]<=0 && temp[i].flag != 1)
        {
                temp[i].flag = 1;
                temp[i].wt = (tcurr+1) - temp[i].bt - temp[i].at;
                temp[i].ta = (tcurr+1) - temp[i].at;
```

```
                sumw+=temp[i].wt;
                sumt+=temp[i].ta;
            }
            j=i;
            min_pr = 999;
            for(x=0;x<n;x++){

                if(temp[x].at <= (tcurr+1) && temp[x].flag != 1){

                    if(min_pr != temp[x].prt && min_pr > temp[x].prt){
                        min_pr = temp[x].prt;
                        i=x;
                    }
                }
            }

        }
        printf(" %d",tcurr);
        avgwt = (float)sumw/n;
        avgta = (float)sumt/n;
        printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);
}


int main(){

        clrscr();
        processes P[10];
        int ch,n;
```

```
do{
        printf("\n\n SIMULATION OF CPU SCHEDULING ALGORITHMS\n");
        printf("\n Options:");
        printf("\n 0. Enter process data.");
        printf("\n 1. FCFS");
        printf("\n 2. SJF (Pre-emptive)");
        printf("\n 3. SJF (Non Pre-emptive)");
        printf("\n 4. Priority Scheduling (Pre-emptive)");
        printf("\n 5. Priority Scheduling (Non Pre-emptive)");
        printf("\n 6. Round Robin");
        printf("\n 7. Exit\n Select : ");
        scanf("%d",&ch);
        switch(ch){
                case 0:
                        n=accept(P);
                        break;
                case 1:
                        FCFS(P,n);
                        break;
                case 2:
                        SJF_P(P,n);
                        break;
                case 3:
                        SJF_NP(P,n);
                        break;
                case 4:
                        PRT_P(P,n);
                        break;
                case 5:
```

```
                    PRT_NP(P,n);
                    break;
                case 6:
                    RR(P,n);
                    break;
                case 7:exit(0);
            }
        }while(ch != 7);
        getch();
        return 0;
}
```

OUTPUT: