Aim:

To simulate First Come First Serve Basis Scheduling Algorithm, Shortest Job First Algorithm and Shortest Remaining Time First Algorithm

Algorithm:

1.***FCFS***:

1.Maintatin a 2d array with arrival time,burst time as columns

2.Declare a min function to find the index where the min arrival time is

3.Swap the index to the 1st Row and viceversa.

4.consider the swapped row till I as sorted

5. After n iterations we obtain a sorted order of PID int arrival time .

6.Compute the completion time for every iteration in the loop

2.***SJF*** :

1. Maintatin a 2d array with pid, arrival time,burst time as columns

2.Declare a function to check if all pid executed

3. Find the minimum arrival time in the 2d array

4.Based on its completion time compare other pid if they arrive within the completion time .

5.Next find the subsequent pid with smallest burst time.

6.While finding those keep track of all other parameters like TAT,CT etc…

3.***SRTF***:

1.In Shortest Remaining Time First first sort the 2d array based on their arrival time

2. Let the first process execute if there is another process that pre-empts the running process wherein its burst time is lesser than latter then prempt the latter executing process

Then reduce the burst time of latter by k units where k is the time it has finished executing

3. Keep track of new pid in an array.

4.Once all pids have been executed atleast once that guarentees that there exist no other process in the pipeline

5.Execute Remaining Process based off their shortest burst time.

Code:

FCFS.c

```
	#include <stdio.h>
#include <stdlib.h>
/*
Implementation of FCFS Scheduling
Works for Even Delay Arrival Process
Usage Whatever you begin with is your serialised ordered process ie.. first one is P1
you can give whatever burst time and arrival time it should be good to go
Usage
no of proc,
arrival_time ,burst time
.
.
.
.
till no_proc

*/
void swap(int *a ,int *b){
	int tmp=*a;
	*a=*b;
	*b=tmp;
}


int min(int proc[][2],int start,int arrlen){
	int mins =proc[start][0];
	int i;
	int minidx;
```

```
        for(i=start;i<arrlen;i++){
                if(proc[i][0]<=mins){
                        minidx=i;
                        mins=proc[i][0];
                }
        }
        return minidx;

}



int main(){
        int num,bt,at;
        printf("Enter the number of Processes:");
        scanf("%d",&num);
        int proctable[num][2];
        int dup[num][2];
        for(int i=0;i<num;i++){

                scanf("%d %d",&at ,&bt);
                proctable[i][0]=at;
                proctable[i][1]=bt;
                dup[i][0]=at;
                dup[i][1]=bt;
        }
        int  idx;
        int idxarr[num];
        int ct=0;
        float c_avg=0.0,tat_avg=0.0,wt_avg=0.0;
        for (int i=0;i<num;i++){
```

```
            idx=min(proctable,i,num);
            idxarr[ct++]=idx;
            swap(&proctable[i][0],&proctable[idx][0]);
            swap(&proctable[i][1],&proctable[idx][1]);
        }
        int ct1=0,k,tat=0,wt=0;

        for(int i=0;i<num;i++){
            printf("%d %d\n",dup[idxarr[i]][0],dup[idxarr[i]][1]);
            if(dup[idxarr[i]][0]>ct){
                k=dup[idxarr[i]][0]-ct+dup[idxarr[i]][1];
                ct1+=k;
                c_avg+=(float)ct1;
            }
            else{
                k=dup[idxarr[i]][1];
                ct1+=k;
                c_avg+=(float)ct1;
            }
            tat=ct1-dup[idxarr[i]][0];
            wt=tat-dup[idxarr[i]][1];
            tat_avg+=(float)tat;
            wt_avg+=(float)wt;
            printf( "\nCompletion Time : %d\t Total Average Time: %d\t Waiting time:
%d\n",ct1,tat,wt);
            swap(&dup[i][0],&dup[idxarr[i]][0]);
            swap(&dup[i][1],&dup[idxarr[i]][1]);
        }
        printf("Average Completion Time: %.2f\nAverage Total Arrival Time:%.2f\nAverage
WaitingTime:%.2f\n",c_avg/(float)num,tat_avg/(float)num,wt_avg/(float)num);
}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise5# ./a.out
Enter the number of Processes:5
0 4
1 3
2 1
3 2
4 5
0 4

Completion Time : 4      Total Average Time: 4   Waiting time: 0
1 3

Completion Time : 7      Total Average Time: 6   Waiting time: 3
2 1

Completion Time : 8      Total Average Time: 6   Waiting time: 5
3 2

Completion Time : 10     Total Average Time: 7   Waiting time: 5
4 5

Completion Time : 15     Total Average Time: 11  Waiting time: 6
Average Completion Time: 8.80
Average Total Arrival Time:6.80
Average WaitingTime:3.80
```

SJF.c

```
#include <stdio.h>
#include <stdlib.h>
int is_vis(int *vis,int pid,int vislen){
        for(int i=0;i<vislen;i++){if(pid==vis[i]){return 1;}}
        return 0;
}
int main(){
        int num,bt,at,pid;
        printf("Enter the number of Processes:");
        scanf("%d",&num);
        int proctable[num][3];
        for(int i=0;i<num;i++){
```

```
        scanf("%d %d %d",&pid,&at ,&bt);
        proctable[i][0]=pid;
        proctable[i][1]=at;
        proctable[i][2]=bt;
}
int visited_pid[num],n;
int min=proctable[0][1],v=0,ct=0,k=0,minidx,tat,wt;
float ct_avg=0.0,tat_avg=0.0,wt_avg=0.0;

for(int j=0;j<num;j++){
                if(proctable[j][1]<=min&& is_vis(visited_pid,proctable[j][0],v)==0){
                        min=proctable[j][1];

                        minidx=j;
                }
        }
visited_pid[v++]=proctable[minidx][0];
ct+=proctable[minidx][2];
tat=ct-proctable[minidx][1];
wt=tat-proctable[minidx][2];
ct_avg+=(float)ct;
tat_avg+=(float)tat;
wt_avg+=(float)wt;
printf("PID:%d \t CT:%d\t TAT:%d\t WT:%d\n",proctable[minidx][0],ct,tat,wt);
while(v<num){
        min=10000;
        for( n=0;n<num;n++){
                if(proctable[n][1]<ct&&is_vis(visited_pid,proctable[n][0],v)==0 ){
                                if(proctable[n][2]<min){
                                        min=proctable[n][2];
                                        minidx=n;
```

```
                                        }
                                }
                        }
                        visited_pid[v++]=proctable[minidx][0];
                        ct+=proctable[minidx][2];
                        tat=ct-proctable[minidx][1];
                        wt=tat-proctable[minidx][2];
                        ct_avg+=(float)ct;
                        tat_avg+=(float)tat;
                        wt_avg+=(float)wt;
                        printf("PID:%d \t CT:%d\t TAT:%d\t WT:%d\n",proctable[minidx][0],ct,tat,wt);
                }
                printf("Average CT :%.2f \t Average TAT:%.2f \t Average WT:%.2f
\n",ct_avg/(float)num,tat_avg/(float)num,wt_avg/(float)num);
}
```

Output:

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise5# ./a.out
Enter the number of Processes:5
1 2 6
2 5 2
3 1 8
4 0 3
5 4 4
PID:4    CT:3    TAT:3   WT:0
PID:1    CT:9    TAT:7   WT:1
PID:2    CT:11   TAT:6   WT:4
PID:5    CT:15   TAT:11  WT:7
PID:3    CT:23   TAT:22  WT:14
Average CT :12.20        Average TAT:9.80        Average WT:5.20
```

3.***SRTF.c***

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

void swap(int *a ,int *b){
	int tmp=*a;
	*a=*b;
	*b=tmp;
}

int is_vis(int *vis,int pid,int vislen){
	for(int i=0;i<vislen;i++){if(pid==vis[i]){return 1;}}
	return 0;
}

void sort(int proc[][3],int len){
	for(int i=0;i<len;i++){
		for(int j=0;j<len-i-1;j++){
			if(proc[j][1]>proc[j+1][1]){
				swap(&proc[j][0],&proc[j+1][0]);
				swap(&proc[j][1],&proc[j+1][1]);
				swap(&proc[j][2],&proc[j+1][2]);
			}
		}
	}

}

int is_burst_zero(int proc[][3],int len){
	int rem_len=0;
	for(int i=0;i<len;i++){
		if(proc[i][2]!=0){
```

```
                        rem_len++;
                }
        }
        if(rem_len==0){
                return 0;
        }
        return 1;

}



int main(){
        int num,bt,at,pid;
        printf("Enter the number of Processes:");
        scanf("%d",&num);
        int proctable[num][3],n,min=proctable[0][1],minidx;
        int dup[num][3];
        int vis[num];
        for(int i=0;i<num;i++){

                scanf("%d %d %d",&pid,&at ,&bt);
                dup[i][0]=pid;
                dup[i][1]=at;
                dup[i][2]=bt;
                proctable[i][0]=pid;
                proctable[i][1]=at;
                proctable[i][2]=bt;
        }
        sort(proctable,num);
        sort(dup,num);
        int proj_c=0;
```

```
/*
printf("\n");
for(int i=0;i<num;i++){
                printf("%d %d %d \n",proctable[i][0],proctable[i][1],proctable[i][2]);
}
*/
int v=0;
int arr[num*num];
int i=0,f=0;
int ct=0,c_proj=0;
int g=0;
int zc=0;
int ct1[num*num];
while(zc<num){
        //printf("\n%d\n",ct);
        //printf("PID:%d \t CT :%d \t TAT:%d\t WT:%d \n",proctable[i][0],ct-proctable[i][1]);
        if(proctable[i][2]==0){
                ct1[f++]=ct;
                zc++;
                i++;
                continue;
        }
        if(i==num){
                i=0;
                continue;
        }
        if(is_vis(vis,proctable[i][0],v)==1){
                ct1[f++]=ct;
                ct+=proctable[i][2];
                proctable[i][2]=0;
```

```
                arr[g++]=proctable[i][0];
                i++;
                continue;
            }
            if(proctable[i+1][2]<proctable[i][2]){
                ct+=proctable[i+1][1]-proctable[i][1];
                proctable[i][2]--;
                arr[g++]=proctable[i][0];
                vis[v++]=proctable[i][0];
            }
            else{
                ct1[f++]=ct;
                ct+=proctable[i][2];
                arr[g++]=proctable[i][0];
                proctable[i][2]=0;
            }
            i++;
        }
        //printf("%d",ct);


        for(int i=0;i<g;i++){
            printf("PID:%d \t CT :%d \t TAT:%d\t WT:%d \n",arr[i],ct1[i],ct1[i]-
dup[arr[i]][1],(ct1[i]-dup[arr[i]][1])-dup[arr[i]][2]);

        }


}
```

Output :

```
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise5# gcc SRTF.c
root@LAPTOP-FHHEGJQ5:/mnt/e/oslab/ajay21110103/Exercise5# ./a.out
Enter the number of Processes:3
1 0 8
2 1 4
3 2 2
```

```
PID:3    CT :11          TAT:11  WT:11
PID:1    CT :14          TAT:13  WT:9
PID:2    CT :14          TAT:12  WT:10
```

Result:

Thus the above given algorithms have been implemented and simulated in c and the output was verified .