ST. XAVIER’S COLLEGE

(Affiliated to Tribhuvan University)
Maitighar, Kathmandu

OS Lab Assignment #6

Scheduling Algorithm

SUBMITTED BY:

Milan Rawal

018BSCIT0

2nd Year/4th Sem

SUBMITTED TO:

Er. Rajan Karmacharya (Coordinator)	
Er Rabin Maharjan (Lecturer)	

Department of Computer Science

STATEMENT: WRITE THE PROGRAM TO IMPLEMENT THE SCHEDULING ALGORITHM USING THE C LANGUAGE.

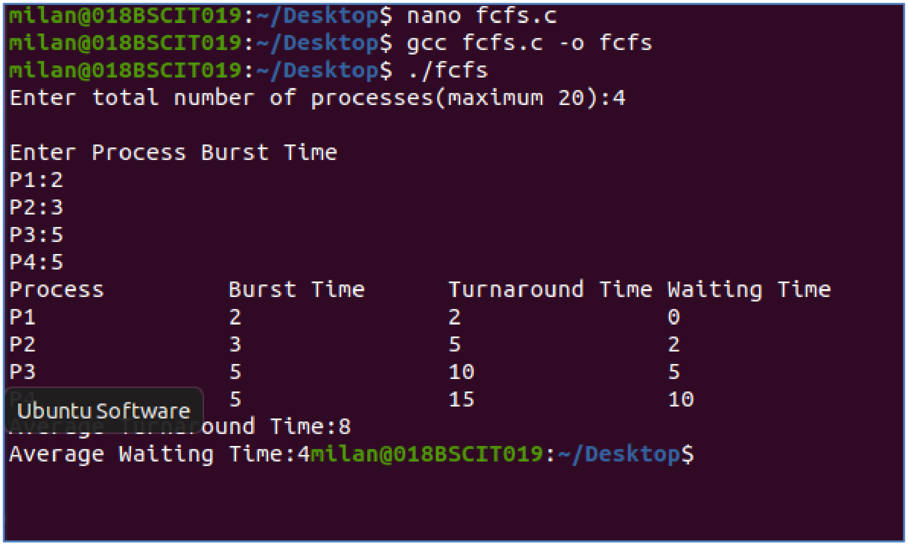
Question: Implement the scheduling algorithm for the following given data by using First Come First Serve, Shortest Job First, Shortest Remaining job First, Round Robin and Priority scheduling algorithms.

1. First come First Serve(FCFS):

Source code:

```
#include<stdio.h>
int main(void){
        int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
        printf("Enter total number of processes(maximum 20):");
        scanf("%d",&n);
        printf("\nEnter Process Burst Time\n");
        for(i=0;i<n;i++){
                printf("P%d:",i+1); scanf("%d",&bt[i]);
        }
        wt[0]=0;
        for(i=1;i<n;i++){
                wt[i]=0;
                for(j=0;j<i;j++) wt[i]+=bt[j];
        }
        printf("Process\t\tBurst Time\tTurnaround Time\tWaiting Time");
        for(i=0;i<n;i++){
                tat[i]=bt[i]+wt[i];
                avwt+=wt[i]; avtat+=tat[i];
                printf("\nP%d\t\t%d\t\t%d\t\t%d",i+1,bt[i],tat[i],wt[i]);
        }
        avwt/=i; avtat/=i;
        printf("\nAverage Turnaround Time:%d",avtat);
        printf("\nAverage Waiting Time:%d",avwt);
}
```

Implementation Screenshot:



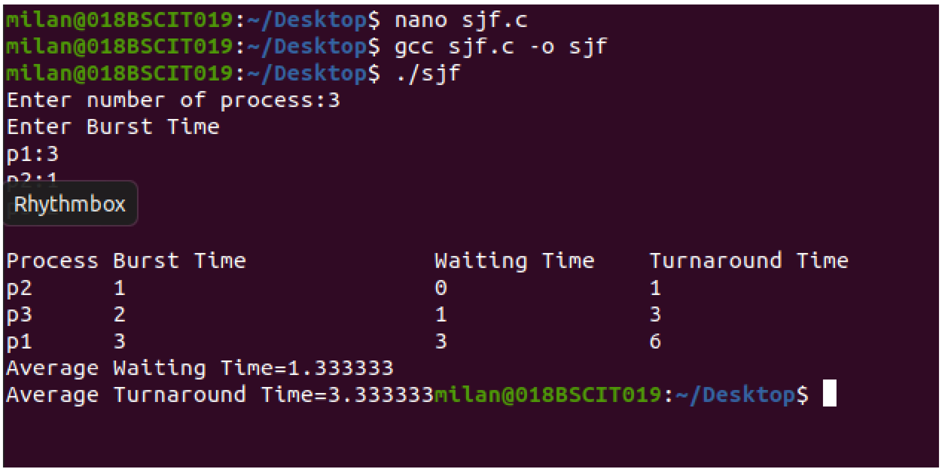
```
milan@018BSCIT019:~/Desktop$ nano fcfs.c
milan@018BSCIT019:~/Desktop$ gcc fcfs.c -o fcfs
milan@018BSCIT019:~/Desktop$ ./fcfs
Enter total number of processes(maximum 20):4

Enter Process Burst Time
P1:2
P2:3
P3:5
P4:5
Process         Burst Time      Turnaround Time Waiting Time
P1              2               2               0
P2              3               5               2
P3              5               10              5
P4              5               15              10
Average Turnaround Time:8
Average Waiting Time:4milan@018BSCIT019:~/Desktop$
```

2. Shortest Job First

Source code:

```
#include<stdio.h>
int main(void){
        int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
        float avg_wt,avg_tat;
        printf("Enter number of process:"); scanf("%d",&n);
        printf("Enter Burst Time\n");
        for(i=0;i<n;i++){
                printf("p%d:",i+1); scanf("%d",&bt[i]); p[i]=i+1;
        }
        for(i=0;i<n;i++){
                pos=i;
                for(j=i+1;j<n;j++) if(bt[j]<bt[pos]) pos=j;
                temp=bt[i]; bt[i]=bt[pos]; bt[pos]=temp; temp=p[i]; p[i]=p[pos]; p[pos]=temp;
        }
        wt[0]=0;
        for(i=1;i<n;i++){
                wt[i]=0;
                for(j=0;j<i;j++) wt[i]+=bt[j];
                total+=wt[i];
        }
        avg_wt=(float)total/n; total=0;
        printf("\nProcess\tBurst Time \tWaiting Time\tTurnaround Time");
        for(i=0;i<n;i++){
                tat[i]=bt[i]+wt[i]; total+=tat[i];
                printf("\np%d\t%d\t\t\t%d\t\t%d",p[i],bt[i],wt[i],tat[i]);
        }
        avg_tat=(float)total/n;
        printf("\nAverage Waiting Time=%f",avg_wt);
        printf("\nAverage Turnaround Time=%f",avg_tat);
}
```

Implementation of the code



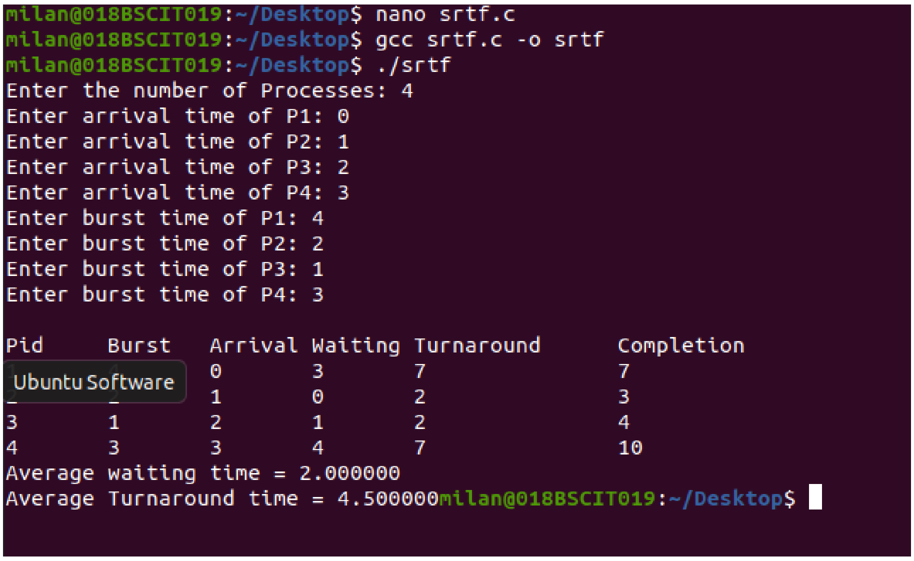
```
milan@018BSCIT019:~/Desktop$ nano sjf.c
milan@018BSCIT019:~/Desktop$ gcc sjf.c -o sjf
milan@018BSCIT019:~/Desktop$ ./sjf
Enter number of process:3
Enter Burst Time
p1:3
p2:1

Process Burst Time              Waiting Time    Turnaround Time
p2      1                       0               1
p3      2                       1               3
p1      3                       3               6
Average Waiting Time=1.333333
Average Turnaround Time=3.333333milan@018BSCIT019:~/Desktop$
```

3. Shortest Remaining Time First

Source code:

```
#include<stdio.h>
int main(void){
        int a[10], b[10], x[10], wt[10], tat[10], ct[10], i,j,min,count=0,time,n;
        double avg=0,tt=0,end;
        printf("Enter the number of Processes: "); scanf("%d",&n);
        for(i=0;i<n;i++){
                printf("Enter arrival time of P%d: ",i+1); scanf("%d",&a[i]);
        }
        for(i=0;i<n;i++){
                printf("Enter burst time of P%d: ",i+1); scanf("%d",&b[i]);
        }
        for(i=0;i<n;i++) x[i]=b[i];
        b[9]=9999;
        for(time=0;count!=n;time++){
                min=9;
                for(i=0;i<n;i++) if(a[i]<=time && b[i]<b[min] && b[i]>0) min=i;
                b[min]--;
                if(b[min]==0){
                        count++; end=time+1; ct[min]=end;
                        wt[min]=end-a[min]-x[min]; tat[min]=end-a[min];
                }
        }
        printf("\nPid\tBurst\tArrival\tWaiting\tTurnaround\tCompletion");
        for(i=0;i<n;i++){
                printf("\n%d\t%d\t%d\t%d\t%d\t\t%d",i+1,x[i],a[i],wt[i],tat[i],ct[i]);
                avg+=wt[i]; tt+=tat[i];
        }
        printf("\nAverage waiting time = %lf\n",avg/n);
        printf("Average Turnaround time = %lf",tt/n);
}
```

Screenshot of the implementation:



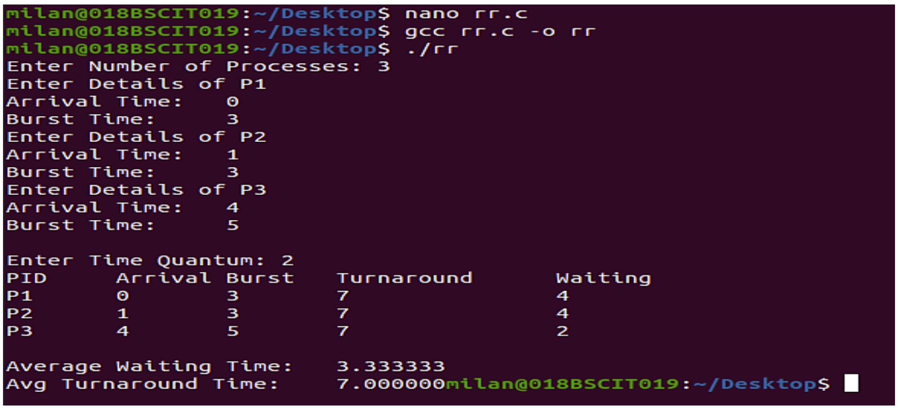
```
milan@018BSCIT019:~/Desktop$ nano srtf.c
milan@018BSCIT019:~/Desktop$ gcc srtf.c -o srtf
milan@018BSCIT019:~/Desktop$ ./srtf
Enter the number of Processes: 4
Enter arrival time of P1: 0
Enter arrival time of P2: 1
Enter arrival time of P3: 2
Enter arrival time of P4: 3
Enter burst time of P1: 4
Enter burst time of P2: 2
Enter burst time of P3: 1
Enter burst time of P4: 3

Pid     Burst   Arrival Waiting Turnaround      Completion
1       4       0       3       7               7
2       2       1       0       2               3
3       1       2       1       2               4
4       3       3       4       7               10
Average waiting time = 2.000000
Average Turnaround time = 4.500000milan@018BSCIT019:~/Desktop$
```

4. Round Robin Problem

Source code:

```
#include<stdio.h>
int main(void){
        int i, total = 0, x, counter = 0, TQ;
        int wt = 0, tat = 0, at[10], bt[10], temp[10];
        float limit, awt, atat;
        printf("Enter Number of Processes: "); scanf("%f", &limit);
        x = limit;
        for(i = 0;i < limit;i++) {
                printf("Enter Details of P%d\n", i + 1);
                printf("Arrival Time:\t"); scanf("%d", &at[i]);
                printf("Burst Time:\t"); scanf("%d", &bt[i]);
                temp[i] = bt[i];
}
        printf("\nEnter Time Quantum: "); scanf("%d", &TQ);
        printf("PID\tArrival\tBurst\tTurnaround\tWaiting");
        for(total = 0, i = 0;x != 0;){
                if(temp[i] <= TQ && temp[i] > 0){
                        total+=temp[i]; temp[i]=0; counter=1;
                } else if(temp[i] > 0){
                        temp[i]-=TQ; total+=TQ;
                }
                if(temp[i] == 0 && counter == 1){
                        printf("\nP%d\t%d\t%d\t%d\t\t%d", i + 1, at[i], bt[i], total - at[i], total - at[i] - bt[i]);
                        x--; wt+=total-at[i]-bt[i];
                        tat+=total-at[i]; counter=0;
                }
                if(i == limit - 1) i = 0;
                else if(at[i + 1] <= total) i++;
                else i = 0;
        }
        awt=wt/limit; atat=tat/limit;
        printf("\n\nAverage Waiting Time:\t%f", awt);
        printf("\nAvg Turnaround Time:\t%f", atat);
}
```

Screenshot of implementation:



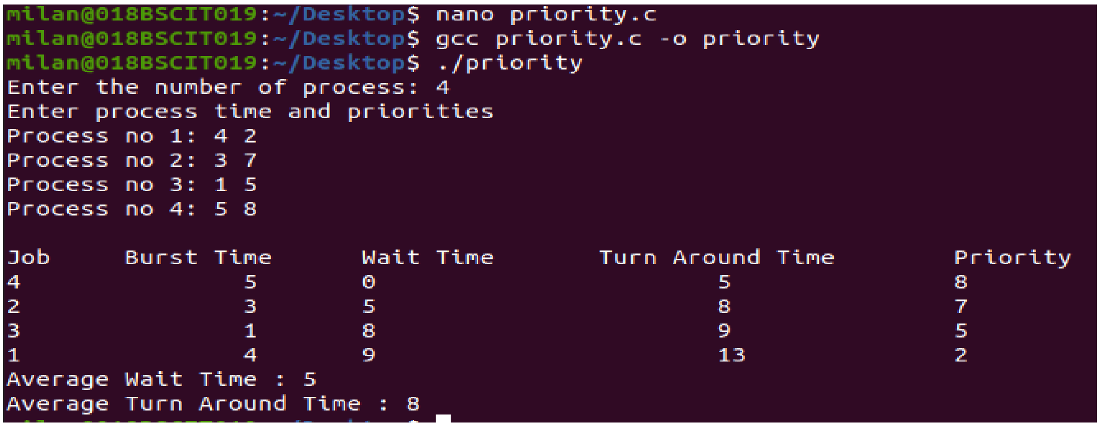
```
milan@018BSCIT019:~/Desktop$ nano rr.c
milan@018BSCIT019:~/Desktop$ gcc rr.c -o rr
milan@018BSCIT019:~/Desktop$ ./rr
Enter Number of Processes: 3
Enter Details of P1
Arrival Time:   0
Burst Time:     3
Enter Details of P2
Arrival Time:   1
Burst Time:     3
Enter Details of P3
Arrival Time:   4
Burst Time:     5

Enter Time Quantum: 2
PID     Arrival Burst   Turnaround      Waiting
P1      0       3       7               4
P2      1       3       7               4
P3      4       5       7               2

Average Waiting Time:   3.333333
Avg Turnaround Time:    7.000000milan@018BSCIT019:~/Desktop$
```

5. Priority scheduling

Source code:

```
#include<stdio.h>
int main(void){
        int x,n,p[10],pp[10],pt[10],w[10],t[10],awt,atat,i;
        printf("Enter the number of process: "); scanf("%d",&n);
        printf("Enter process time and priorities\n");
        for(i=0;i<n;i++){
                printf("Process no %d: ",i+1);
                scanf("%d %d",&pt[i],&pp[i]);
                p[i]=i+1;
        }
        for(i=0;i<n-1;i++)
                for(int j=i+1;j<n;j++)
                        if(pp[i]<pp[j]){
                                x=pp[i]; pp[i]=pp[j]; pp[j]=x;
                                x=pt[i]; pt[i]=pt[j]; pt[j]=x;
                                x=p[i]; p[i]=p[j]; p[j]=x;
                        }
        w[0]=0; awt=0; t[0]=pt[0]; atat=t[0];
        for(i=1;i<n;i++){
                w[i]=t[i-1]; awt+=w[i];
                t[i]=w[i]+pt[i]; atat+=t[i];
        }
        printf("\nJob\tBurst Time\tWait Time\tTurn Around Time\tPriority\n");
        for(i=0;i<n;i++)
                printf("%d\t\t%d\t%d\t\t\t%d\t\t%d\n",p[i],pt[i],w[i],t[i],pp[i]);
        awt/=n; atat/=n;
        printf("Average Wait Time : %d\n",awt);
        printf("Average Turn Around Time : %d\n",atat);
}
```

Screenshot of implementation:



```
milan@018BSCIT019:~/Desktop$ nano priority.c
milan@018BSCIT019:~/Desktop$ gcc priority.c -o priority
milan@018BSCIT019:~/Desktop$ ./priority
Enter the number of process: 4
Enter process time and priorities
Process no 1: 4 2
Process no 2: 3 7
Process no 3: 1 5
Process no 4: 5 8

Job     Burst Time      Wait Time       Turn Around Time        Priority
4               5       0                       5               8
2               3       5                       8               7
3               1       8                       9               5
1               4       9                       13              2
Average Wait Time : 5
Average Turn Around Time : 8
```

Comparison: So, from these all solutions we see that we can achieve the best average time by using Shortest Remaining Time First.

Conclusion:

So, we can implement different scheduling algorithms using C-program.