Name – Ashutosh Ardu

Regno – 20BRS1262

OS Lab 6

Scheduling Algorithm

Code

(The Code could be pretty huge)

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

struct job{
    int pid,burst,arrival,priority,wait,TAT,rt;
};

void WaitingTime_fcfs(struct job fcfs[], int n,int wt[]){
    wt[0] = 0;
    for (int i = 1; i < n ; i++ )
        wt[i] = fcfs[i-1].burst + wt[i-1] ;
}
void TurnAroundTime_fcfs( struct job fcfs[], int n, int wt[], int tat[]){
    for (int i = 0; i < n ; i++)
        tat[i] = fcfs[i].burst+ wt[i];
}

void avgTime_fcfs(struct job fcfs[],int n)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    WaitingTime_fcfs(fcfs, n,wt);
    TurnAroundTime_fcfs(fcfs, n, wt, tat);
    printf("Process\tBurstTime\tWaitingTime\tTurnAroundTime\n");
    for (int i=0; i<n; i++){
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf("%d\t",(i+1));
        printf("%d\t\t", fcfs[i].burst );
        printf("%d\t\t",wt[i] );
        printf("%d\t\n",tat[i]);
    }
```

```
float s=(float)total_wt / (float)n;
    float t=(float)total_tat / (float)n;
    printf("Average waiting time = %f\n",s);
    printf("Average turn around time = %f\n",t);
}

void SJF(struct job sjf[],int n){
    int wt[20],tat[20],i,j,total=0,pos,temp;
    float avg_wt,avg_tat;
    for(i=0;i<n;i++){
        pos=i;
        for(j=i+1;j<n;j++){
            if(sjf[j].burst<sjf[pos].burst) pos=j;
        }temp=sjf[i].burst;
        sjf[i].burst=sjf[pos].burst;
        sjf[pos].burst=temp;
        temp=sjf[i].pid;
        sjf[i].pid=sjf[pos].pid;
        sjf[pos].pid=temp;
    }
    wt[0]=0;
    for(i=1;i<n;i++){
        wt[i]=0;
        for(j=0;j<i;j++) wt[i]+=sjf[j].burst;
        total+=wt[i];
    }avg_wt=(float)total/n;
    total=0;
    printf("Process\tBurstTime\tWaitingTime\tTurnAroundTime\n");
    for(i=0;i<n;i++){
        tat[i]=sjf[i].burst+wt[i];
        total+=tat[i];
        printf("%d\t%d\t\t%d\t\t%d\n",sjf[i].pid,sjf[i].burst,wt[i],tat[i
]);
    }avg_tat=(float)total/n;
    printf("Average Waiting Time=%f\n",avg_wt);
    printf("Average Turnaround Time=%f\n",avg_tat);
}

void WaitingTime_srjf(struct job proc[], int n,int wt[]){
    int rt[n];
    for (int i = 0; i < n; i++) rt[i] = proc[i].burst;
    int complete = 0, t = 0, minm = 100000;
    int shortest = 0, finish_time;
    int check = 0;
    while (complete != n) {
        for (int j = 0; j < n; j++) {
            if ((proc[j].arrival <= t) &&
            (rt[j] < minm) && rt[j] > 0) {
                minm = rt[j];
                shortest = j;
                check = 1;
            }
```

```
  }
    if (!check) {
      t++;
      continue;
    }
    rt[shortest]--;minm = rt[shortest];
    if (minm == 0) minm = 10000;
    if (rt[shortest] == 0) {
      complete++;
      check = 0;
      finish_time = t + 1;
      wt[shortest] = finish_time -proc[shortest].burst -
proc[shortest].arrival;
      if (wt[shortest] < 0) wt[shortest] = 0;
    }
    t++;
  }
}

void TurnAroundTime_srjf(struct job proc[], int n,int wt[], int tat[]){
  for (int i = 0; i < n; i++) tat[i] = proc[i].burst + wt[i];
}
void avgTime_srjf(struct job proc[], int n){
  int wt[n], tat[n], total_wt = 0,total_tat = 0;
  WaitingTime_srjf(proc, n, wt);
  TurnAroundTime_srjf(proc, n, wt, tat);
 printf("Process\tBurstTime\tWaitingTime\tTurnAroundTime\n");
  for (int i = 0; i < n; i++) {
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
  printf("%d\t%d\t\t%d\t\t%d\n",proc[i].pid,proc[i].burst,wt[i],tat[i])
;
  }printf("Average Waiting Time %f\n",((float)total_wt/(float)n));
 printf("Average Turn Around Time %f\n",((float)total_tat/(float)n));
}

void swap_pnp(struct job *a,struct job *b){
 int t=a->pid;
 a->pid=b->pid;b->pid=t;
 t=a->burst;a->burst=b->burst;
 b->burst=t;
 t=a->priority;a->priority=b->priority;
 b->priority=t;
}

void WaitingTime_pnp(struct job proc[], int n,int wt[]){
  wt[0] = 0;
  for (int i = 1; i < n ; i++ )
    wt[i] = proc[i-1].burst + wt[i-1] ;
}

void TurnAroundTime_pnp(struct  job proc[], int n,int wt[], int tat[]){
  for (int i = 0; i < n ; i++)
    tat[i] = proc[i].burst + wt[i];
}

void avgTime_pnp(struct job proc[], int n){
  int wt[n], tat[n], total_wt = 0, total_tat = 0;
  WaitingTime_pnp(proc, n, wt);
  TurnAroundTime_pnp(proc, n, wt, tat);
 printf("Process\tBurstTime\tWaitingTime\tTurnAroundTime\n");
  for (int i=0; i<n; i++){
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
  printf("%d\t%d\t\t%d\t\t%d\n",proc[i].pid,proc[i].burst,wt[i],tat[i])
;
  }printf("Average waiting time = %f\n",((float)total_wt/(float)n));
 printf("Average turn around time = %f\n",((float)total_tat/(float)n));
}

void priorityScheduling_pnp(struct job proc[], int n){
 int minm;
  for(int i=0;i<n-1;++i){
  minm=i;
  for(int j=i+1;j<n;++j){
   if(proc[j].priority<proc[minm].priority) minm=j;
  }swap_pnp(&proc[i],&proc[minm]);
 }
 printf("Order in which processes gets executed \n");
  for (int i = 0 ; i < n; i++) printf("%d ",proc[i].pid);
 printf("\n");
  avgTime_pnp(proc, n);
}

void pp(struct job a[],int n){
 int temp[10],t,count=0,short_p;
 float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
 for(int i=0;i<n;i++) temp[i]=a[i].burst;
  a[9].priority=10000;
  for(t=0;count!=n;t++){
    short_p=9;
    for(int i=0;i<n;i++){
      if(a[short_p].priority>a[i].priority && a[i].arrival<=t && a[
i].burst>0) short_p=i;
    }a[short_p].burst=a[short_p].burst-1;
    if(a[short_p].burst==0){
      count++;
      a[short_p].wait=t+1-a[short_p].arrival-temp[short_p];
      a[short_p].TAT=t+1-a[short_p].arrival;
      total_WT=total_WT+a[short_p].wait;
      total_TAT=total_TAT+a[short_p].TAT;

    }
  }Avg_WT=total_WT/n;
```

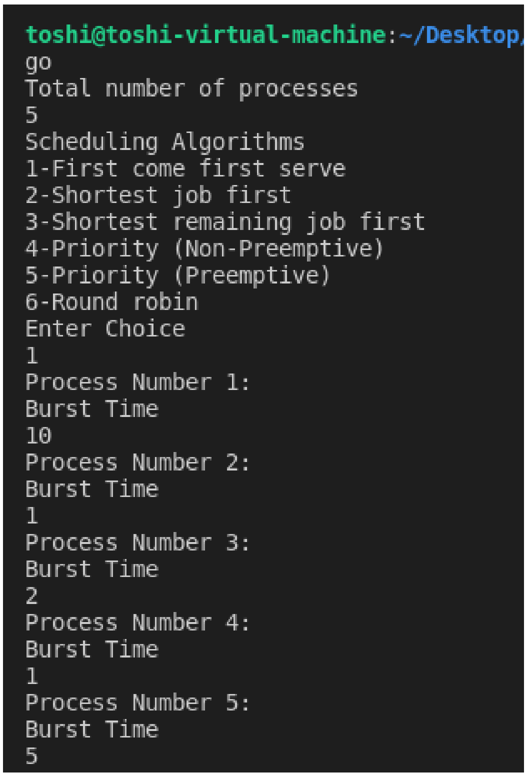
```
    Avg_TAT=total_TAT/n;
    printf("ID\tAT\tBT\tP\tWT\tTAT\n");
    for(int i=0;i<n;i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",i+1,a[i].arrival,temp[i],a[i].priority,a[i].wait,a[i].TAT);
    printf("Avg waiting time of the process  is %f\n",Avg_WT);
    printf("Avg turn around time of the process is %f\n",Avg_TAT);
}

void RR(struct job rr[],int n){
  int count,j,time,remain,flag=0,time_quantum;
    float wait_time=0,turnaround_time=0;
    remain=n;
    for(count=0;count<n;count++) rr[count].rt=rr[count].burst;
    printf("Enter Time Quantum:\n");
    scanf("%d",&time_quantum);
    printf("\n\nProcess\t|Turnaround Time|Waiting Time\n");
    for(time=0,count=0;remain!=0;){
        if(rr[count].rt<=time_quantum && rr[count].rt>0)
        {
        time+=rr[count].rt;
        rr[count].rt=0;
        flag=1;
        }
        else if(rr[count].rt>0)
        {
        rr[count].rt-=time_quantum;
        time+=time_quantum;
        }
        if(rr[count].rt==0 && flag==1)
        {
        remain--;
        printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-rr[count].arrival,time-rr[count].arrival-rr[count].burst);
        wait_time+=time-rr[count].arrival-rr[count].burst;
        turnaround_time+=time-rr[count].arrival;
        flag=0;
        }
        if(count==n-1)
        count=0;
        else if(rr[count+1].arrival<=time)
        count++;
        else
        count=0;
    }
    printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
    printf("Avg Turnaround Time = %f\n",turnaround_time*1.0/n);
}

int main()
{
    int ch,n;
    printf("Total number of processes\n");
    scanf("%d",&n);
    struct job fcfs[n],sjf[n],srjf[n],rr[n],priority[n];
    printf("Scheduling Algorithms\n1-First come first serve\n2-
Shortest job first\n3-Shortest remaining job first\n4-Priority (Non-
Preemptive)\n5-Priority (Preemptive)\n6-Round robin\n");
    printf("Enter Choice\n");
    scanf("%d",&ch);
    switch(ch){
        case 1:
            for(int i=0;i<n;++i){
                printf("Process Number %d:\n",i+1);
                fcfs[i].pid=i+1;
                printf("Burst Time\n");
                scanf("%d",&fcfs[i].burst);
            }avgTime_fcfs(fcfs,n);
            break;
        case 2:
            for(int i=0;i<n;++i){
                printf("Process Number %d:\n",i+1);
                sjf[i].pid=i+1;
                printf("Burst Time\n");
                scanf("%d",&sjf[i].burst);
            }SJF(sjf,n);
            break;
        case 3:
            for(int i=0;i<n;++i){
                printf("Process Number %d:\n",i+1);
                srjf[i].pid=i+1;
                printf("Burst Time\n");
                scanf("%d",&srjf[i].burst);
                printf("Arrival Time\n");
                scanf("%d",&srjf[i].arrival);
            }avgTime_srjf(srjf,n);
            break;
        case 4:
            for(int i=0;i<n;++i){
                printf("Process Number %d:\n",i+1);
                priority[i].pid=i+1;
                printf("Burst Time\n");
                scanf("%d",&priority[i].burst);
                printf("Priority\n");
                scanf("%d",&priority[i].priority);
            }priorityScheduling_pnp(priority,n);
            break;
        case 5:
            for(int i=0;i<n;++i){
                printf("Process Number %d:\n",i+1);
                priority[i].pid=i+1;
                printf("Arrival Time\n");
                scanf("%d",&priority[i].arrival);
                printf("Burst Time\n");
                scanf("%d",&priority[i].burst);
                printf("Priority\n");
                scanf("%d",&priority[i].priority);
            }pp(priority,n);
            break;
        case 6:
          for(int i=0;i<n;++i){
            printf("Process Number %d:\n",i+1);
            rr[i].pid=i+1;
            printf("Arrival & Burst Time\n");
            scanf("%d %d",&rr[i].arrival,&rr[i].burst);
          }RR(rr,n);
            break;
        default:
            exit(0);
    }
    return 0;
}
```

Output

The Initial Gantt's Charts for all the Algorithms

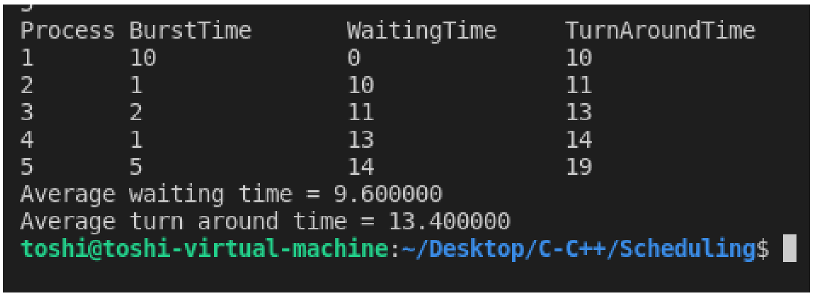
S.no	Process ID	Arrival time	Expected Burst time	Priority
1	P1	0	10	3
2	P2	2	1	1
3	P3	4	2	4
4	P4	7	1	5
5	P5	3	5	2

First Come First Serve Algorithm



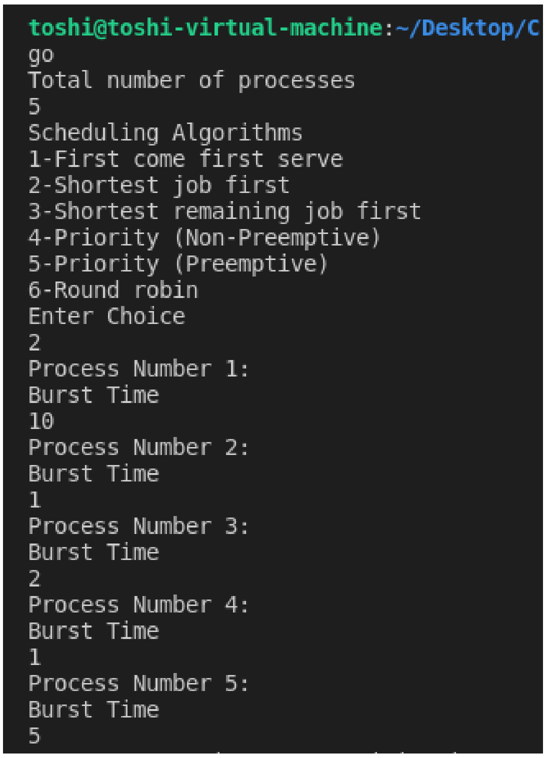
```
toshi@toshi-virtual-machine:~/Desktop/
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
1
Process Number 1:
Burst Time
10
Process Number 2:
Burst Time
1
Process Number 3:
Burst Time
2
Process Number 4:
Burst Time
1
Process Number 5:
Burst Time
5
```

Final Gantt’s Chart



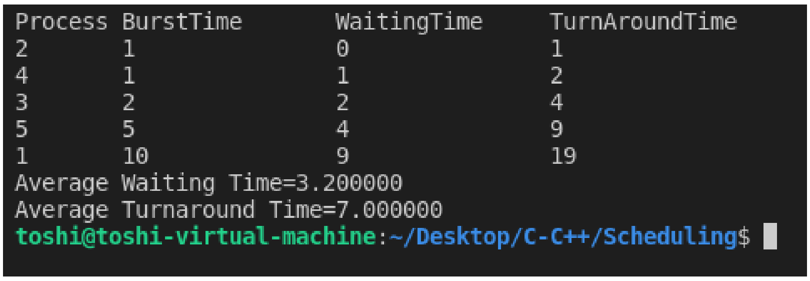
```
Process BurstTime       WaitingTime     TurnAroundTime
1       10              0               10
2       1               10              11
3       2               11              13
4       1               13              14
5       5               14              19
Average waiting time = 9.600000
Average turn around time = 13.400000
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$ 
```

Shortest Job First Algorithm



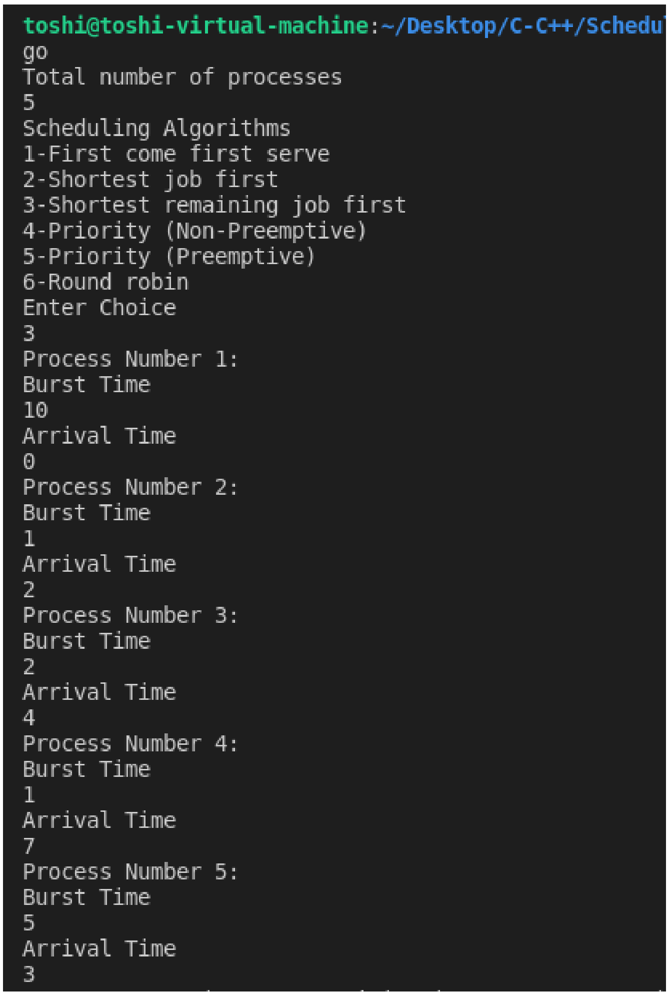
```
toshi@toshi-virtual-machine:~/Desktop/C
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
2
Process Number 1:
Burst Time
10
Process Number 2:
Burst Time
1
Process Number 3:
Burst Time
2
Process Number 4:
Burst Time
1
Process Number 5:
Burst Time
5
```

Final Gantt’s Chart



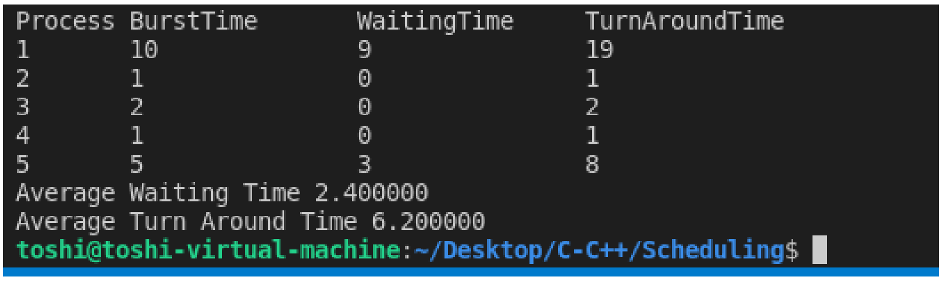
```
Process BurstTime       WaitingTime     TurnAroundTime
2       1               0               1
4       1               1               2
3       2               2               4
5       5               4               9
1       10              9               19
Average Waiting Time=3.200000
Average Turnaround Time=7.000000
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$ 
```

Shortest Remaining Job First Algorithm



```
toshi@toshi-virtual-machine:~/Desktop/C-C++/Schedu
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
3
Process Number 1:
Burst Time
10
Arrival Time
0
Process Number 2:
Burst Time
1
Arrival Time
2
Process Number 3:
Burst Time
2
Arrival Time
4
Process Number 4:
Burst Time
1
Arrival Time
7
Process Number 5:
Burst Time
5
Arrival Time
3
```

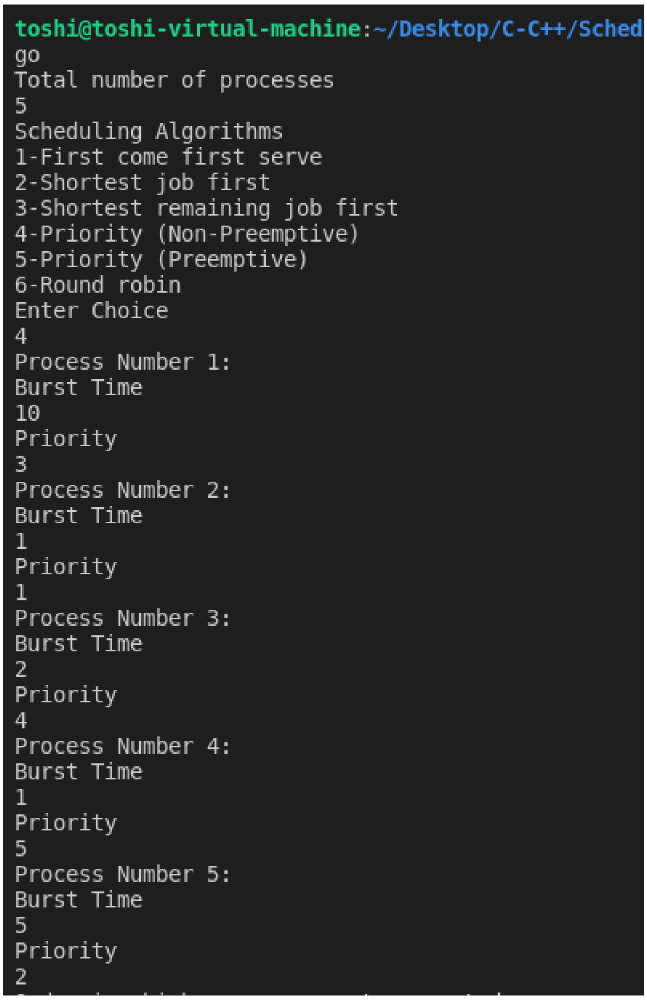
Final Gantt's Chart



```
Process BurstTime       WaitingTime     TurnAroundTime
1       10              9               19
2       1               0               1
3       2               0               2
4       1               0               1
5       5               3               8
Average Waiting Time 2.400000
Average Turn Around Time 6.200000
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$
```

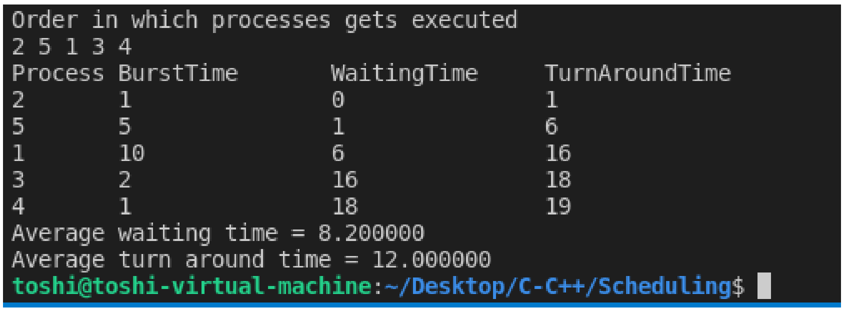
Priority Scheduling Algorithm

(Non-Preemptive)



```
toshi@toshi-virtual-machine:~/Desktop/C-C++/Sched
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
4
Process Number 1:
Burst Time
10
Priority
3
Process Number 2:
Burst Time
1
Priority
1
Process Number 3:
Burst Time
2
Priority
4
Process Number 4:
Burst Time
1
Priority
5
Process Number 5:
Burst Time
5
Priority
2
```

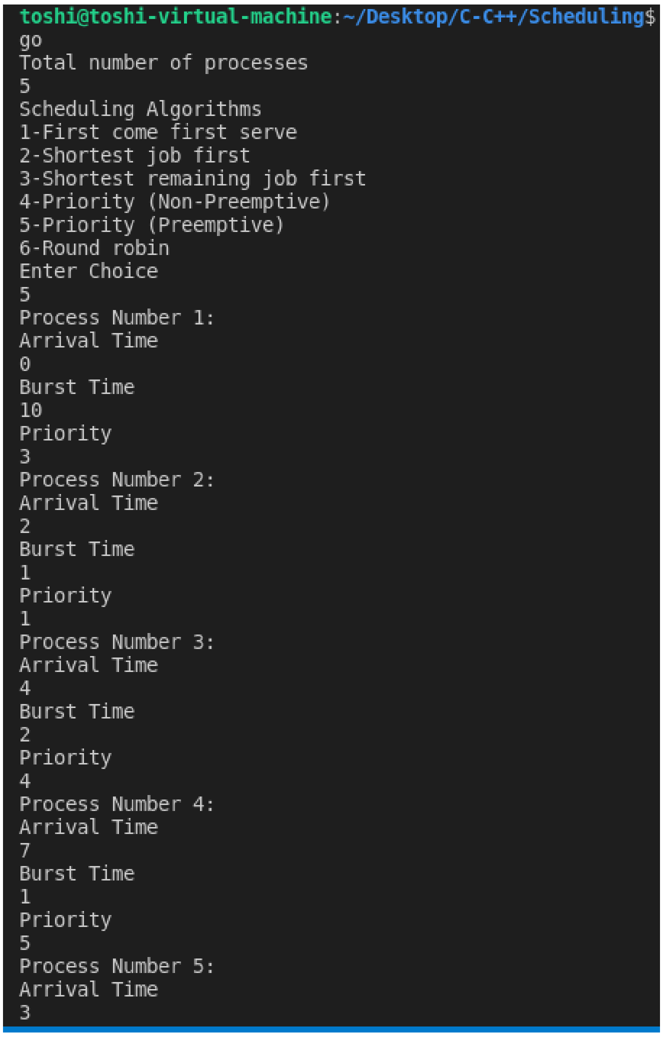
Final Gantt’s Chart



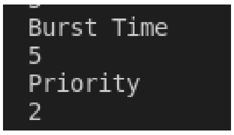
```
Order in which processes gets executed
2 5 1 3 4
Process BurstTime       WaitingTime     TurnAroundTime
2       1               0               1
5       5               1               6
1       10              6               16
3       2               16              18
4       1               18              19
Average waiting time = 8.200000
Average turn around time = 12.000000
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$
```

Priority Scheduling Algorithm

(Preemptive)

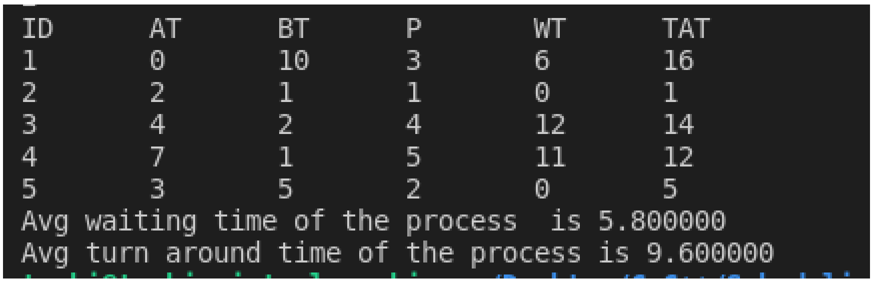


```
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
5
Process Number 1:
Arrival Time
0
Burst Time
10
Priority
3
Process Number 2:
Arrival Time
2
Burst Time
1
Priority
1
Process Number 3:
Arrival Time
4
Burst Time
2
Priority
4
Process Number 4:
Arrival Time
7
Burst Time
1
Priority
5
Process Number 5:
Arrival Time
3
```



```
Burst Time
5
Priority
2
```

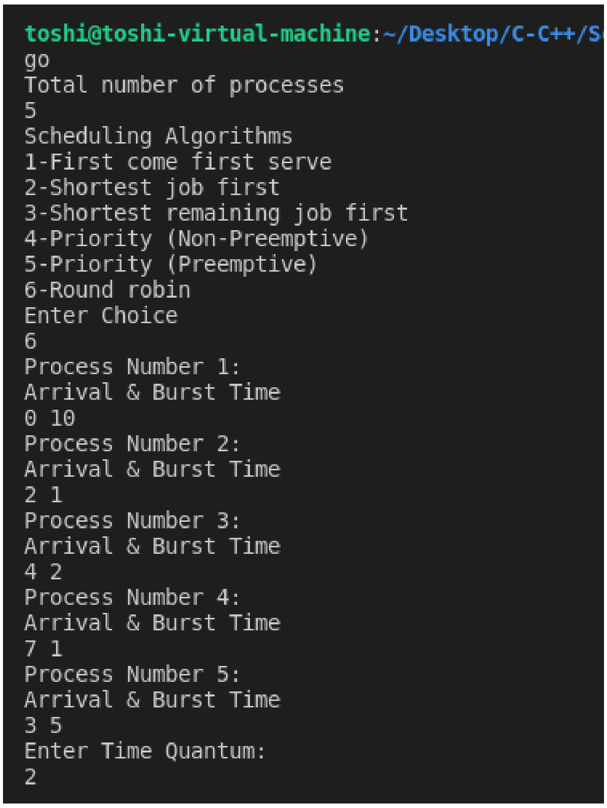
Final Gantt’s Chart



```
ID        AT        BT        P         WT        TAT
1         0         10        3         6         16
2         2         1         1         0         1
3         4         2         4         12        14
4         7         1         5         11        12
5         3         5         2         0         5
Avg waiting time of the process  is 5.800000
Avg turn around time of the process is 9.600000
```

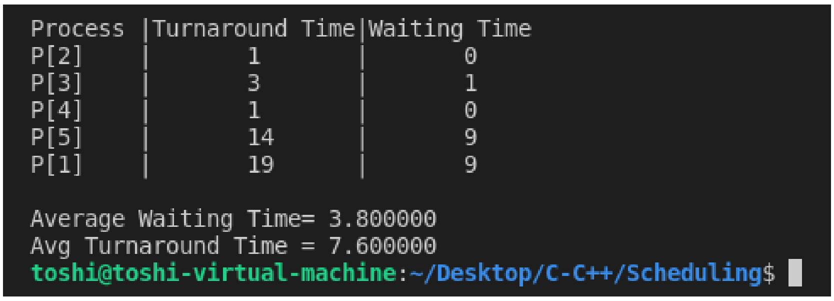
Round Robin

(Preemptive)



```
toshi@toshi-virtual-machine:~/Desktop/C-C++/S
go
Total number of processes
5
Scheduling Algorithms
1-First come first serve
2-Shortest job first
3-Shortest remaining job first
4-Priority (Non-Preemptive)
5-Priority (Preemptive)
6-Round robin
Enter Choice
6
Process Number 1:
Arrival & Burst Time
0 10
Process Number 2:
Arrival & Burst Time
2 1
Process Number 3:
Arrival & Burst Time
4 2
Process Number 4:
Arrival & Burst Time
7 1
Process Number 5:
Arrival & Burst Time
3 5
Enter Time Quantum:
2
```

Final Gantt's Chart



```
Process |Turnaround Time|Waiting Time
P[2]    |       1       |       0
P[3]    |       3       |       1
P[4]    |       1       |       0
P[5]    |       14      |       9
P[1]    |       19      |       9

Average Waiting Time= 3.800000
Avg Turnaround Time = 7.600000
toshi@toshi-virtual-machine:~/Desktop/C-C++/Scheduling$ 
```

Conclusion

Scheduling Algorithm	Waiting Time	Total Turn Around Time
Round Robin	3.8 units	7.6 units
Priority (Preemptive)	5.8 units	9.6 units
Priority(Non-Preemptive)	8.2 units	12.0 units
Shortest Remaining Time First	2.4 units	6.2 units
Shortest Job First	3.2 units	7.0 units
First Come First Serve	9.6 units	13.4 units

Most effective : Shortest Remaining Time First

Least effective : First come first serve