EAST WEST UNIVERSITY

excellence in education

Department of CSE

Report of CPU Scheduling Project

Course name: Operating System

Course code: CSE325

Section: 03

Submitted from,

Iffat Tasnim

ID: 2020-2-60-004

Department of Computer Science and Engineering

Submitted to

Dr. Motaharul Islam

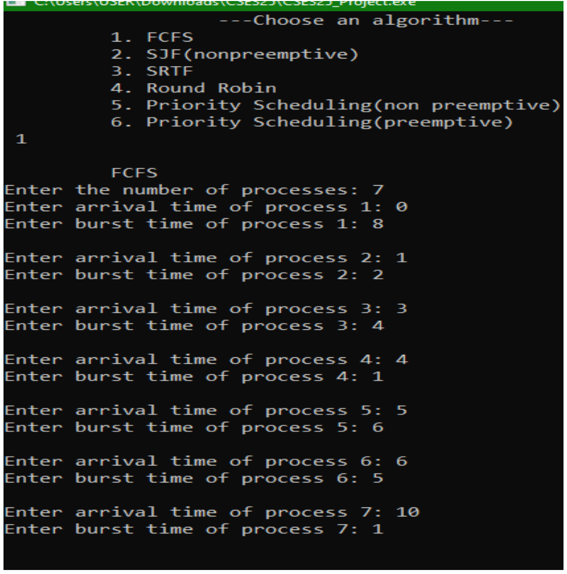
Professor,

Department of Computer Science and Engineering

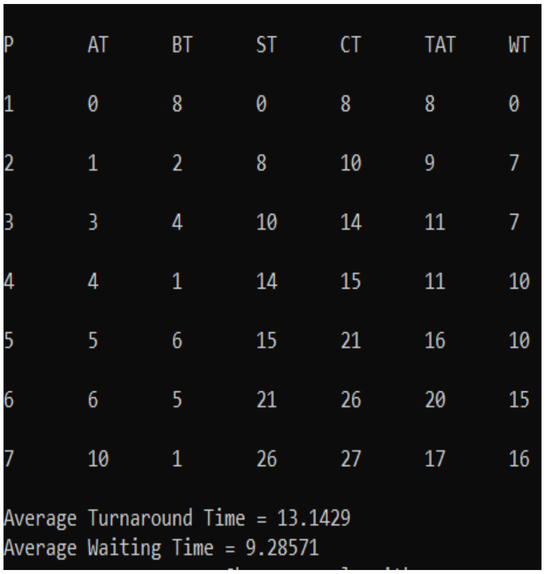
CPU SCHEDULING

FCFS:

```
    int n,  total_turnaround_time = 0,  total_waiting_time = 0; struct process p[100];
    float avg_turnaround_time, avg_waiting_time, avg_response_time;
    cout<<"Enter the number of processes: "; cin>>n;
    for(int i = 0; i < n; i++) {
        cout<<"Enter arrival time of process "<<i+1<<": "; cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
        p[i].pid = i+1; cout<<endl;
    }
    for(int i = 0; i < n; i++) {
        p[i].start_time = (i == 0)?p[i].arrival_time:max(p[i-1].completion_time,p[i].arrival_time);
        p[i].completion_time = p[i].start_time + p[i].burst_time;
        p[i].turnaround_time = p[i].completion_time - p[i].arrival_time;
        p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;
        total_turnaround_time = total_turnaround_time + p[i].turnaround_time;
        total_waiting_time = total_waiting_time + p[i].waiting_time;
    }
    avg_turnaround_time=(float)total_turnaround_time/n; avg_waiting_time =(float)total_waiting_time/n;
    cout<<endl; cout<<"P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++){
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```



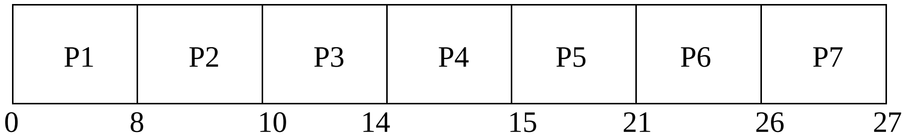
```
                    ---Choose an algorithm---
          1. FCFS
          2. SJF(nonpreemptive)
          3. SRTF
          4. Round Robin
          5. Priority Scheduling(non preemptive)
          6. Priority Scheduling(preemptive)
 1

          FCFS
Enter the number of processes: 7
Enter arrival time of process 1: 0
Enter burst time of process 1: 8

Enter arrival time of process 2: 1
Enter burst time of process 2: 2

Enter arrival time of process 3: 3
Enter burst time of process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1

Enter arrival time of process 5: 5
Enter burst time of process 5: 6

Enter arrival time of process 6: 6
Enter burst time of process 6: 5

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
```



```
P       AT      BT      ST      CT      TAT     WT

1       0       8       0       8       8       0

2       1       2       8       10      9       7

3       3       4       10      14      11      7

4       4       1       14      15      11      10

5       5       6       15      21      16      10

6       6       5       21      26      20      15

7       10      1       26      27      17      16

Average Turnaround Time = 13.1429
Average Waiting Time = 9.28571
```

Criteria- Arrival time, Mode- Non-preemptive

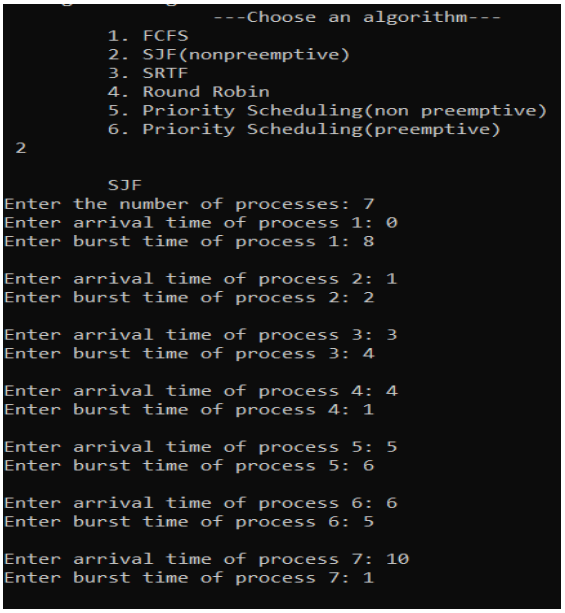
Gantt Chart-



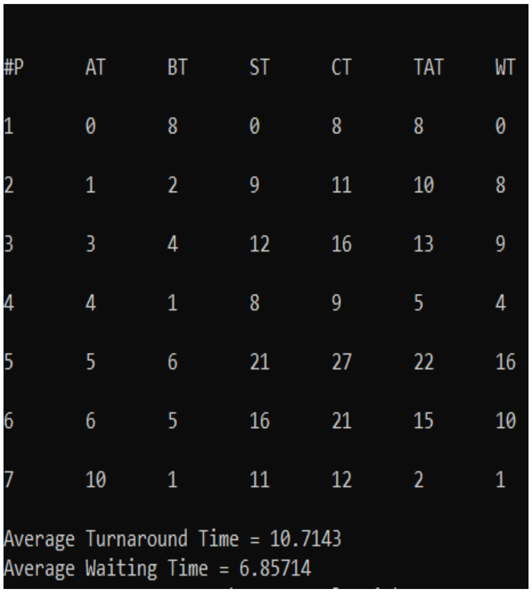
SJF:

```
    int n,total_turnaround_time = 0,total_waiting_time = 0,total_response_time = 0; int is_completed[100];
    struct process p[100]; float avg_turnaround_time, avg_waiting_time, avg_response_time;
    memset(is_completed,0,sizeof(is_completed));
    cout<<"Enter the number of processes: ";cin>>n;
    for(int i = 0; i < n; i++) {
        cout<<"Enter arrival time of process "<<i+1<<": "; cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
        p[i].pid = i+1; cout<<endl;
    }
    int current_time = 0, completed = 0, prev = 0;
    while(completed != n) {
        int idx = -1; int mn = 10000000;
        for(int i = 0; i < n; i++){
            if(p[i].arrival_time <= current_time && is_completed[i] == 0){
                if(p[i].burst_time < mn){
                    mn = p[i].burst_time;  idx = i;
                }
                if(p[i].burst_time == mn){
                    if(p[i].arrival_time < p[idx].arrival_time){
                        mn = p[i].burst_time;  idx = i;
                    }
                }
            }
        }
        if(idx != -1){
            p[idx].start_time = current_time;
            p[idx].completion_time = p[idx].start_time + p[idx].burst_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
            is_completed[idx] = 1;
            completed++;
            current_time = p[idx].completion_time; prev = current_time;
        }
        else{
            current_time++;
        }
    }
    int min_arrival_time = 10000000, max_completion_time = -1;;
    for(int i = 0; i < n; i++){
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time =(float)total_turnaround_time/n; avg_waiting_time=(float)total_waiting_time /n;
```

```
cout<<endl<<endl;cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].start_time<<"\t"<<p[i].comp
letion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```

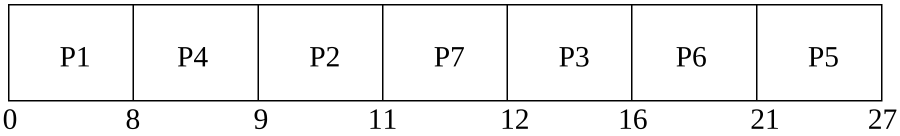


```
                        ---Choose an algorithm---
               1. FCFS
               2. SJF(nonpreemptive)
               3. SRTF
               4. Round Robin
               5. Priority Scheduling(non preemptive)
               6. Priority Scheduling(preemptive)
 2

               SJF
Enter the number of processes: 7
Enter arrival time of process 1: 0
Enter burst time of process 1: 8

Enter arrival time of process 2: 1
Enter burst time of process 2: 2

Enter arrival time of process 3: 3
Enter burst time of process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1

Enter arrival time of process 5: 5
Enter burst time of process 5: 6

Enter arrival time of process 6: 6
Enter burst time of process 6: 5

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
```



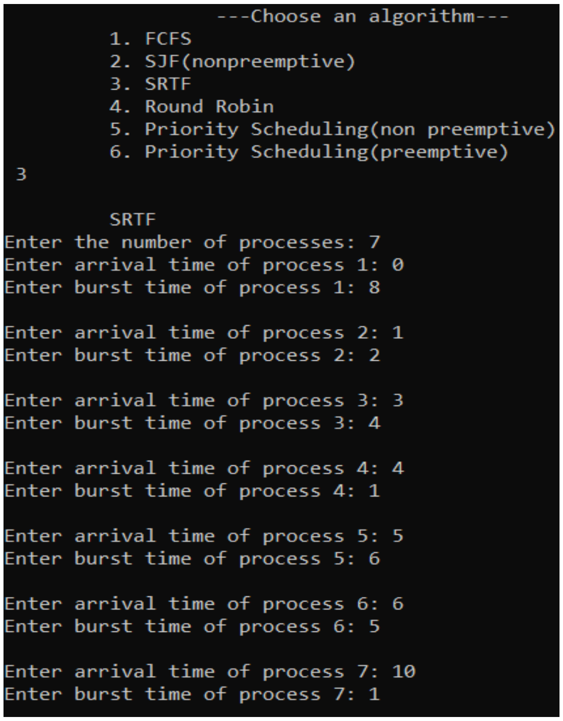
```
#P      AT      BT      ST      CT      TAT     WT

1       0       8       0       8       8       0

2       1       2       9       11      10      8

3       3       4       12      16      13      9

4       4       1       8       9       5       4

5       5       6       21      27      22      16

6       6       5       16      21      15      10

7       10      1       11      12      2       1

Average Turnaround Time = 10.7143
Average Waiting Time = 6.85714
```

Criteria- Burst time, Mode- Non-preemptive
Gantt Chart-

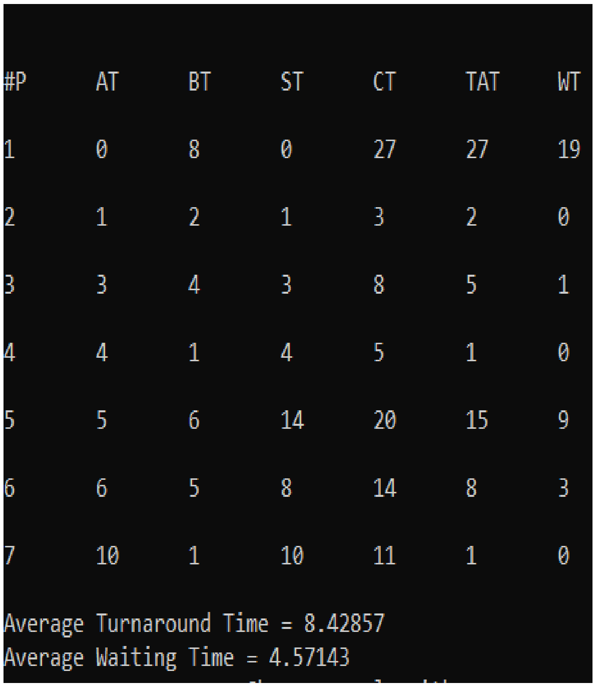


SRTF:

```
    struct process p[100];
    float avg_turnaround_time, avg_waiting_time, avg_response_time;
    int n, total_turnaround_time = 0, total_waiting_time = 0;
    int burst_remaining[100]; int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));
    cout<<"Enter the number of processes: ";  cin>>n;
    for(int i = 0; i < n; i++)  {
        cout<<"Enter arrival time of process "<<i+1<<": "; cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
        p[i].pid = i+1;  burst_remaining[i] = p[i].burst_time; cout<<endl;
    }
    int current_time = 0, completed = 0, prev = 0;
    while(completed != n){
        int idx = -1, mn = 10000000;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0){
                if(burst_remaining[i] < mn){
                    mn = burst_remaining[i];  idx = i;
                }
                if(burst_remaining[i] == mn){
                    if(p[i].arrival_time < p[idx].arrival_time){
                        mn = burst_remaining[i];  idx = i;
                    }
                }
            }
        }
        if(idx != -1){
            if(burst_remaining[idx] == p[idx].burst_time){
                p[idx].start_time = current_time;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;
            if(burst_remaining[idx] == 0){
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;
                is_completed[idx] = 1;
                completed++;
            }
        }
        else {
            current_time++;
        }
    }
    int min_arrival_time = 10000000, max_completion_time = -1;
    for(int i = 0; i < n; i++){
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time = (float)total_turnaround_time/n;avg_waiting_time =(float)total_waiting_time/n;
    cout<<endl<<endl;
    cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].start_time<<"\t"<<p[i].comp
letion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```



```
                ---Choose an algorithm---
        1. FCFS
        2. SJF(nonpreemptive)
        3. SRTF
        4. Round Robin
        5. Priority Scheduling(non preemptive)
        6. Priority Scheduling(preemptive)
 3

        SRTF
Enter the number of processes: 7
Enter arrival time of process 1: 0
Enter burst time of process 1: 8

Enter arrival time of process 2: 1
Enter burst time of process 2: 2

Enter arrival time of process 3: 3
Enter burst time of process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1

Enter arrival time of process 5: 5
Enter burst time of process 5: 6

Enter arrival time of process 6: 6
Enter burst time of process 6: 5

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
```



```
#P      AT      BT      ST      CT      TAT     WT

1       0       8       0       27      27      19

2       1       2       1       3       2       0

3       3       4       3       8       5       1

4       4       1       4       5       1       0

5       5       6       14      20      15      9

6       6       5       8       14      8       3

7       10      1       10      11      1       0

Average Turnaround Time = 8.42857
Average Waiting Time = 4.57143
```

Criteria- Burst time, Mode- preemptive

Gantt Chart-

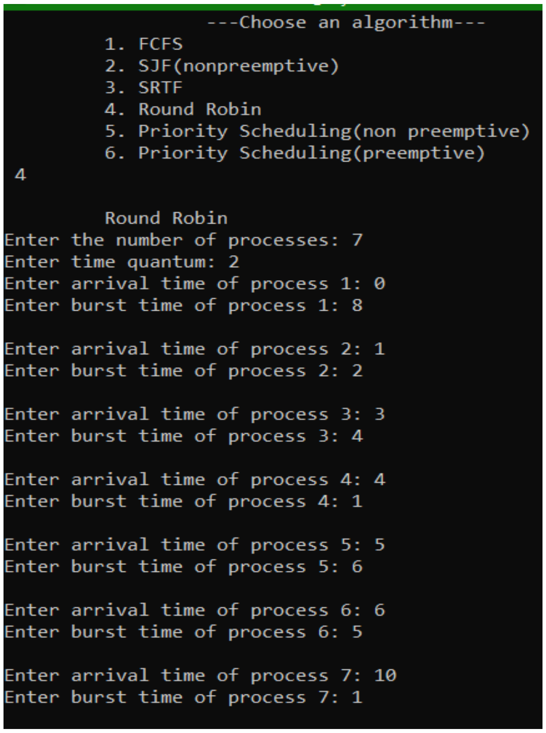
P1	P2	P2	P3	P4	P3	P3	P3	P6	P6	P7	P6	P6	P5	P1

0 1 2 3 4 5 6 7 8 9 10 11 12 14 20 27

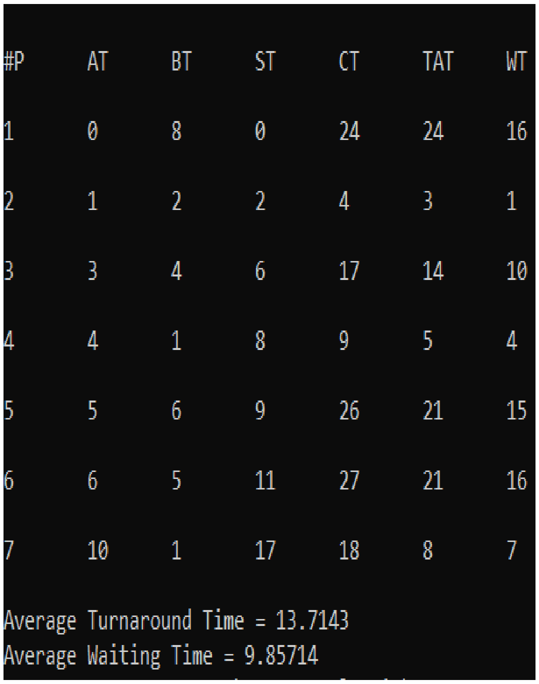
Round Robin:

```
int n, tq, total_turnaround_time = 0, total_waiting_time = 0, idx;
struct process p[100];
float avg_turnaround_time, avg_waiting_time;
int burst_remaining[100];
cout<<"Enter the number of processes: "; cin>>n;
cout<<"Enter time quantum: "; cin>>tq;
for(int i = 0; i < n; i++){
    cout<<"Enter arrival time of process "<<i+1<<": ";  cin>>p[i].arrival_time;
    cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
    burst_remaining[i] = p[i].burst_time; p[i].pid = i+1;
    cout<<endl;
}
sort(p,p+n,compare1);
queue<int> q;
int current_time = 0;
q.push(0);
int completed = 0;
int mark[100];
memset(mark,0,sizeof(mark));
mark[0] = 1;
while(completed != n) {
        idx = q.front();
        q.pop();
        if(burst_remaining[idx] == p[idx].burst_time){
            p[idx].start_time = max(current_time,p[idx].arrival_time);
            current_time = p[idx].start_time;
        }
        if(burst_remaining[idx]-tq > 0) {
            burst_remaining[idx] -= tq;
            current_time += tq;
        }
        else{
            current_time += burst_remaining[idx];
            burst_remaining[idx] = 0;
            completed++;
            p[idx].completion_time = current_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
        }
        for(int i = 1; i < n; i++){
            if(burst_remaining[i] > 0 && p[i].arrival_time <= current_time && mark[i] == 0){
                q.push(i); mark[i] = 1;
            }
        }
        if(burst_remaining[idx] > 0) {
            q.push(idx);
        }
        if(q.empty()){
            for(int i = 1; i < n; i++){
                if(burst_remaining[i] > 0){
                    q.push(i);
                    mark[i] = 1;
                    break;
                }
            }
        }
    }
    avg_turnaround_time =(float) total_turnaround_time/n; avg_waiting_time=(float)total_waiting_time/n;
    sort(p,p+n,compare2);
    cout<<endl;
    cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"RT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++) {
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].start_time<<"\t"<<p[i].comp
letion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<p[i].response_time<<"\t"<<"
\n"<<endl;
```

```
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```



```
                  ---Choose an algorithm---
         1. FCFS
         2. SJF(nonpreemptive)
         3. SRTF
         4. Round Robin
         5. Priority Scheduling(non preemptive)
         6. Priority Scheduling(preemptive)
 4

         Round Robin
Enter the number of processes: 7
Enter time quantum: 2
Enter arrival time of process 1: 0
Enter burst time of process 1: 8

Enter arrival time of process 2: 1
Enter burst time of process 2: 2

Enter arrival time of process 3: 3
Enter burst time of process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1

Enter arrival time of process 5: 5
Enter burst time of process 5: 6

Enter arrival time of process 6: 6
Enter burst time of process 6: 5

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
```



```
#P      AT      BT      ST      CT      TAT     WT

1       0       8       0       24      24      16

2       1       2       2       4       3       1

3       3       4       6       17      14      10

4       4       1       8       9       5       4

5       5       6       9       26      21      15

6       6       5       11      27      21      16

7       10      1       17      18      8       7

Average Turnaround Time = 13.7143
Average Waiting Time = 9.85714
```

Criteria-Arrival time+ time quantum, Mode- preemptive

Ready Queue-

~~P1~~ ~~P2~~ ~~P1~~ ~~P3~~ ~~P4~~ ~~P5~~ ~~P6~~ ~~P1~~ ~~P3~~ ~~P7~~ ~~P5~~ ~~P6~~ ~~P1~~ ~~P5~~ ~~P6~~

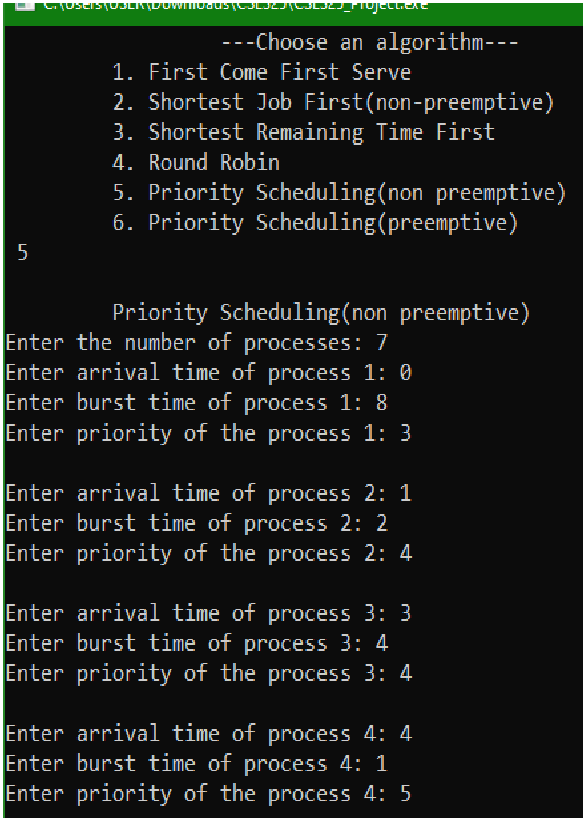
Gantt Chart-

P1	P2	P1	P3	P4	P5	P6	P1	P3	P7	P5	P6	P1	P5	P6

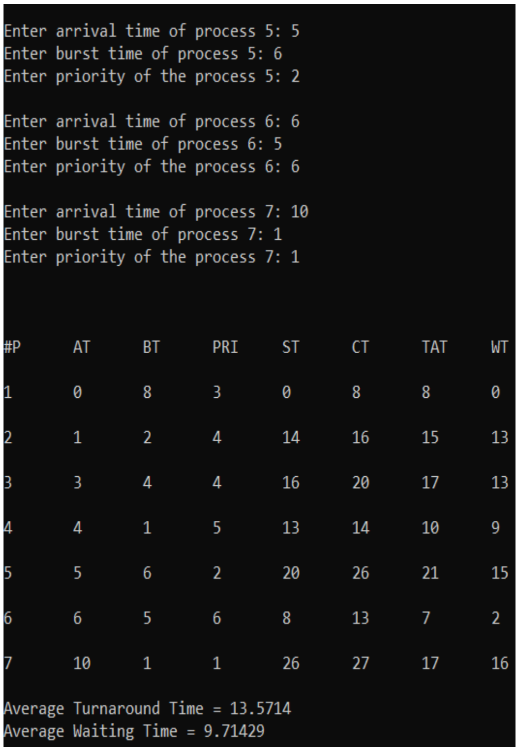
0 2 4 6 8 9 11 13 15 17 18 20 22 24 26 27

Priority Schedule(non-preemptive):

```
    int n, total_turnaround_time = 0, total_waiting_time = 0;
    struct process p[100];
    float avg_turnaround_time, avg_waiting_time;, avg_response_time;
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));
    cout<<"Enter the number of processes: ";
    cin>>n;
    for(int i = 0; i < n; i++){
        cout<<"Enter arrival time of process "<<i+1<<": "; cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
        cout<<"Enter priority of the process "<<i+1<<": "; cin>>p[i].priority;
        p[i].pid = i+1; cout<<endl;
    }
    int current_time = 0, completed = 0, prev = 0;
    while(completed != n){
        int idx = -1; int mx = -1;
        for(int i = 0; i < n; i++){
            if(p[i].arrival_time <= current_time && is_completed[i] == 0){
                if(p[i].priority > mx){
                    mx = p[i].priority;
                    idx = i;
                }
                if(p[i].priority == mx){
                    if(p[i].arrival_time < p[idx].arrival_time){
                        mx = p[i].priority;
                        idx = i;
                    }
                }
            }
        }
        if(idx != -1){
            p[idx].start_time = current_time;
            p[idx].completion_time = p[idx].start_time + p[idx].burst_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
            is_completed[idx] = 1;
            completed++;
            current_time = p[idx].completion_time;
            prev = current_time;
        }
        else{
            current_time++;
        }
    }
    int min_arrival_time = 10000000, max_completion_time = -1;
    for(int i = 0; i < n; i++){
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;
    cout<<endl<<endl;
    cout<<"#P\t"<<"AT\t"<<"BT\t"<<"PRI\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++){
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].priority<<"\t"<<p[i].start_ti
me<<"\t"<<p[i].completion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<"\t"<<"
\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```

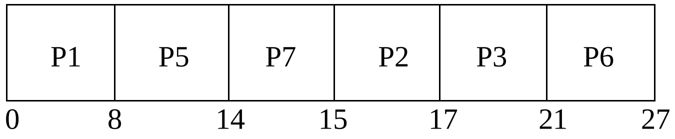


```
C:\Users\USER\Downloads\CSE325\CSE325_Project.exe
               ---Choose an algorithm---
        1. First Come First Serve
        2. Shortest Job First(non-preemptive)
        3. Shortest Remaining Time First
        4. Round Robin
        5. Priority Scheduling(non preemptive)
        6. Priority Scheduling(preemptive)
 5

        Priority Scheduling(non preemptive)
Enter the number of processes: 7
Enter arrival time of process 1: 0
Enter burst time of process 1: 8
Enter priority of the process 1: 3

Enter arrival time of process 2: 1
Enter burst time of process 2: 2
Enter priority of the process 2: 4

Enter arrival time of process 3: 3
Enter burst time of process 3: 4
Enter priority of the process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1
Enter priority of the process 4: 5
```



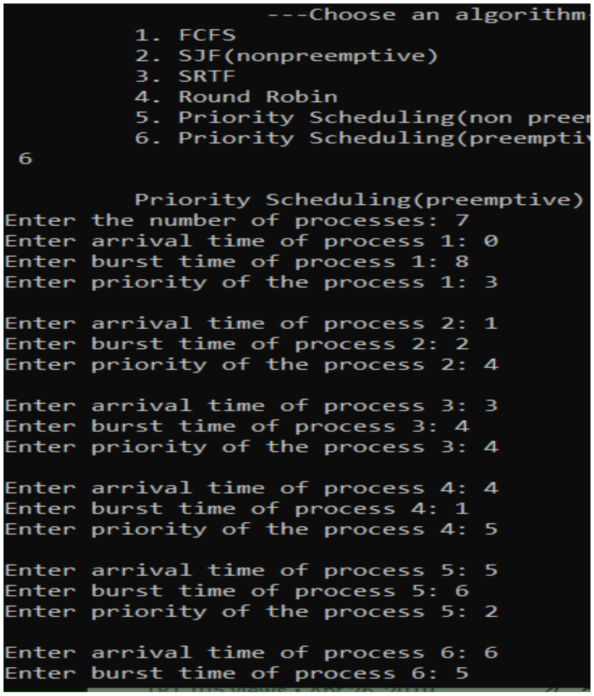
```
Enter arrival time of process 5: 5
Enter burst time of process 5: 6
Enter priority of the process 5: 2

Enter arrival time of process 6: 6
Enter burst time of process 6: 5
Enter priority of the process 6: 6

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
Enter priority of the process 7: 1



#P      AT      BT      PRI     ST      CT      TAT     WT

1       0       8       3       0       8       8       0

2       1       2       4       14      16      15      13

3       3       4       4       16      20      17      13

4       4       1       5       13      14      10      9

5       5       6       2       20      26      21      15

6       6       5       6       8       13      7       2

7       10      1       1       26      27      17      16

Average Turnaround Time = 13.5714
Average Waiting Time = 9.71429
```

Criteria- priority, Mode- non-preemptive
Gantt Chart-

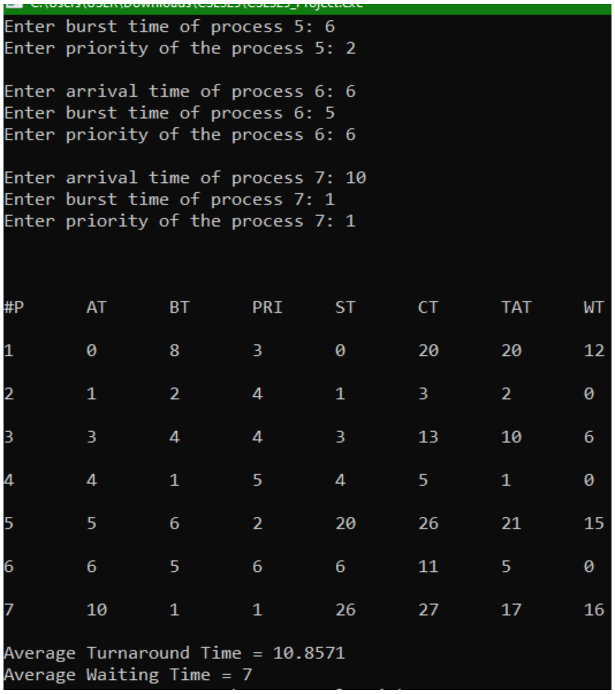


Priority Schedule(pre-emptive):

```
    int n, total_turnaround_time = 0, total_waiting_time = 0 ;
    struct process p[100];
    float avg_turnaround_time, avg_waiting_time, avg_response_time;
    int burst_remaining[100];
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));
    cout<<"Enter the number of processes: ";  cin>>n;
    for(int i = 0; i < n; i++){
        cout<<"Enter arrival time of process "<<i+1<<": ";cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": "; cin>>p[i].burst_time;
        cout<<"Enter priority of the process "<<i+1<<": "; cin>>p[i].priority;
        p[i].pid = i+1; burst_remaining[i] = p[i].burst_time; cout<<endl;
    }
    int current_time = 0, completed = 0, prev = 0;
    while(completed != n){
        int idx = -1, mx = -1;
        for(int i = 0; i < n; i++)  {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0)  {
                if(p[i].priority > mx)   {
                    mx = p[i].priority;
                    idx = i;
                }
                if(p[i].priority == mx) {
                    if(p[i].arrival_time < p[idx].arrival_time)  {
                        mx = p[i].priority;
                        idx = i;
                    }
                }
            }
        }
        if(idx != -1)  {
            if(burst_remaining[idx] == p[idx].burst_time)  {
                p[idx].start_time = current_time;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;
            if(burst_remaining[idx] == 0) {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;
                is_completed[idx] = 1;
                completed++;
            }
        }
        else{
            current_time++;
        }
    }
    int min_arrival_time = 10000000; int max_completion_time = -1;
    for(int i = 0; i < n; i++){
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }
    avg_turnaround_time=(float)total_turnaround_time /n; avg_waiting_time=(float) total_waiting_time/n;
    cout<<endl<<endl;
    cout<<"-P\t"<<"AT\t"<<"BT\t"<<"PRI\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"\n"<<endl;
    for(int i = 0; i < n; i++){
cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].priority<<"\t"<<p[i].start_ti
me<<"\t"<<p[i].completion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<"\t"<<"
\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
```



```
                         ---Choose an algorithm
          1. FCFS
          2. SJF(nonpreemptive)
          3. SRTF
          4. Round Robin
          5. Priority Scheduling(non preem
          6. Priority Scheduling(preempti
 6

          Priority Scheduling(preemptive)
Enter the number of processes: 7
Enter arrival time of process 1: 0
Enter burst time of process 1: 8
Enter priority of the process 1: 3

Enter arrival time of process 2: 1
Enter burst time of process 2: 2
Enter priority of the process 2: 4

Enter arrival time of process 3: 3
Enter burst time of process 3: 4
Enter priority of the process 3: 4

Enter arrival time of process 4: 4
Enter burst time of process 4: 1
Enter priority of the process 4: 5

Enter arrival time of process 5: 5
Enter burst time of process 5: 6
Enter priority of the process 5: 2

Enter arrival time of process 6: 6
Enter burst time of process 6: 5
```



```
Enter burst time of process 5: 6
Enter priority of the process 5: 2

Enter arrival time of process 6: 6
Enter burst time of process 6: 5
Enter priority of the process 6: 6

Enter arrival time of process 7: 10
Enter burst time of process 7: 1
Enter priority of the process 7: 1



#P      AT      BT      PRI     ST      CT      TAT     WT

1       0       8       3       0       20      20      12

2       1       2       4       1       3       2       0

3       3       4       4       3       13      10      6

4       4       1       5       4       5       1       0

5       5       6       2       20      26      21      15

6       6       5       6       6       11      5       0

7       10      1       1       26      27      17      16

Average Turnaround Time = 10.8571
Average Waiting Time = 7
```

Criteria- priority, Mode- preemptive
Gantt Chart-

P1	P1	P1	P1	P5	P5	P7	P5	P1	P2	P3	P4	P6	
0	1	3	4	5	6	10	11	12	15	17	21	22	27

Comparison:

Waiting time is an amount of time. A process in ready queue waits a certain amount of time to enter the CPU and that's called waiting time of the process. Waiting time= Turnaround time – Burst time.
Here,
Turnaround time = The amount of time of a process from arrival time to completion time,
Burst time = The amount of time of a process used for execution,
Average waiting time for all schedules:

FCFS:	9.28571ms
SJF:	6.85714ms
SRTF:	4.57143ms
Round Robin:	9.85714ms
Priority Schedule(non-preemptive):	9.71429ms
Priority Schedule(preemptive):	7ms

Here. SRTF has the less average waiting time than others. It means a process has to wait almost 4.571ms in the ready queue to get CPU which is far better than the other cases. So, SRTF is the best case.