OPERATING SYSTEMS LAB

PRACTICAL 5

NAME: VEDANT BHUTADA

ROLL: 69

BATCH: A4

AIM: Given the list of processes, their CPU burst time, arrival time and time quantum. Display/print the Gantt chart; compute the average waiting time and average turnaround time for the following scheduling policies.

a) Preemptive SJF

b) Round Robin

c) Preemptive Priority

d) Non-preemptive SJF

e) Non-preemptive Priority

f) First Come First Serve

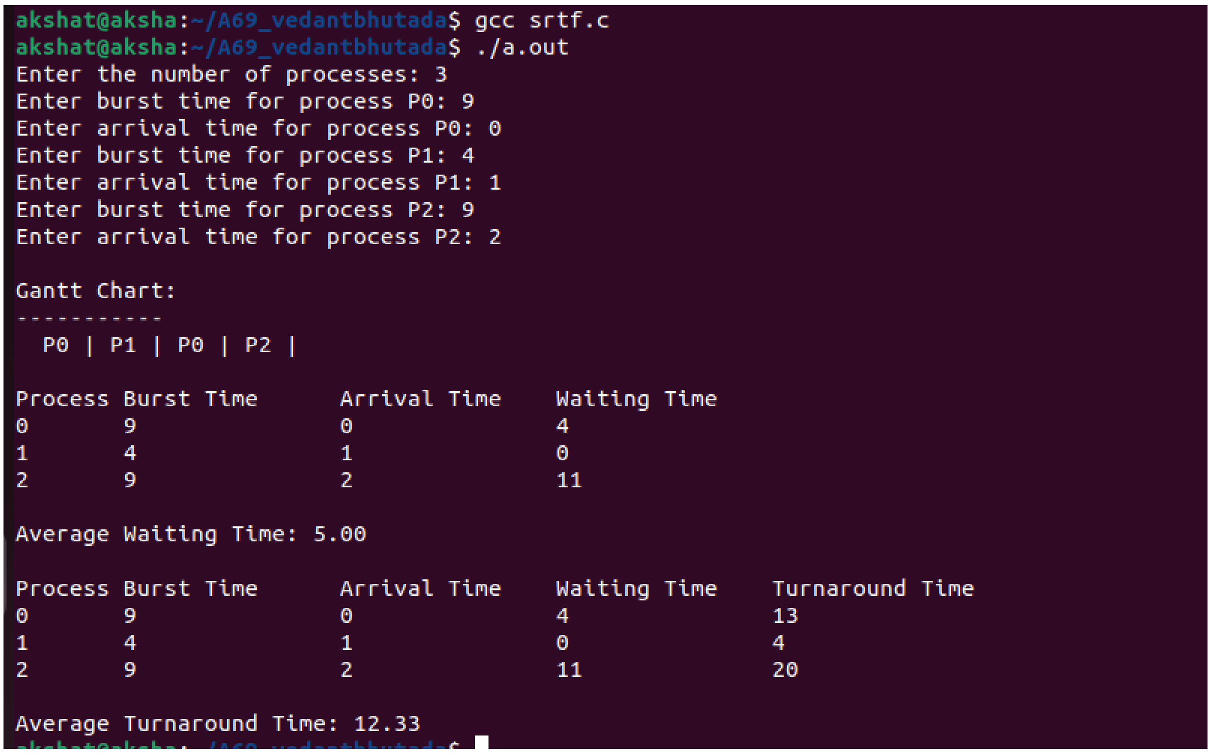
CODE:

a) Preemptive SJF

```
#include <stdio.h>

struct Process {
   int pid;
   int burst_time;
   int arrival_time;
   int remaining_time;
   int waiting_time;
   int turnaround_time;
};

void calculateWaitingTime(struct Process proc[], int n) {
   int i;
   int total_wt = 0;

   for (i = 0; i < n; i++) {
       proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;
       total_wt += proc[i].waiting_time;
```

```
    }

    float avg_wt = (float)total_wt / n;

    printf("\nProcess\tBurst Time\tArrival Time\tWaiting Time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].arrival_time,
proc[i].waiting_time);

    printf("\nAverage Waiting Time: %.2f\n", avg_wt);
}

void calculateTurnaroundTime(struct Process proc[], int n) {
    int i;
    int total_tat = 0;

    for (i = 0; i < n; i++) {
        proc[i].turnaround_time = proc[i].burst_time + proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
    }

    float avg_tat = (float)total_tat / n;

    printf("\nProcess\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].arrival_time,
proc[i].waiting_time, proc[i].turnaround_time);

    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
}

void srtfScheduling(struct Process proc[], int n) {
    int i, j;
    int current_time = 0;
    int completed = 0;

    for (i = 0; i < n; i++)
        proc[i].remaining_time = proc[i].burst_time;

    printf("\nGantt Chart:\n");
    printf("-----------\n");
    printf(" ");
    int current_process = -1;

    while (completed != n) {
        int shortest_job = -1;
        int shortest_time = 9999;

        for (i = 0; i < n; i++) {
            if (proc[i].arrival_time <= current_time && proc[i].remaining_time < shortest_time &&
proc[i].remaining_time > 0) {
```

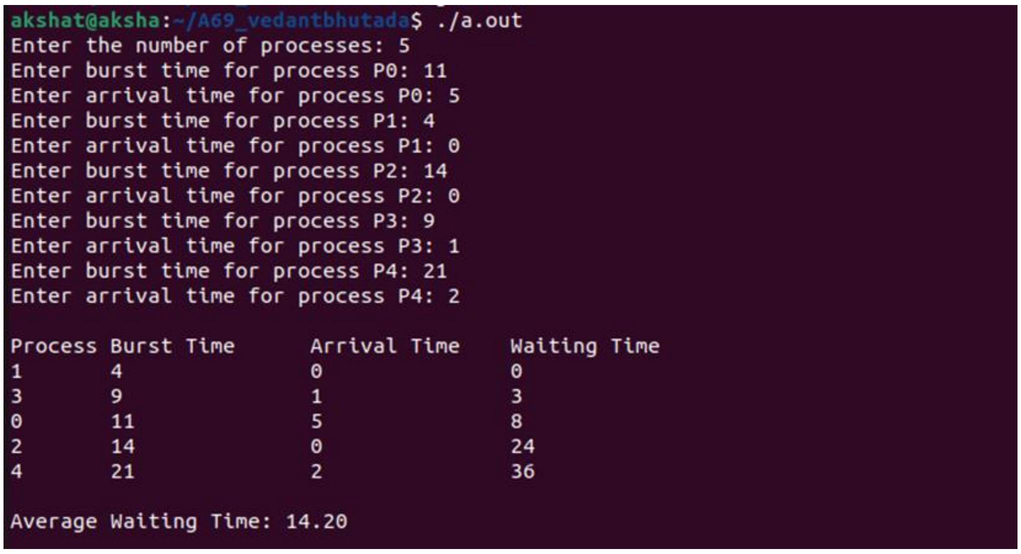
```
                shortest_job = i;
                shortest_time = proc[i].remaining_time;
            }
        }

        if (shortest_job != -1) {
            if (current_process != shortest_job) {
                if (current_process != -1)
                    printf("|");

                printf(" P%d ", proc[shortest_job].pid);
                current_process = shortest_job;
            }

            proc[shortest_job].remaining_time--;
            current_time++;

            if (proc[shortest_job].remaining_time == 0) {
                completed++;
                proc[shortest_job].turnaround_time = current_time - proc[shortest_job].arrival_time;
            }
        } else {
            current_time++;
        }
    }

    printf("|\n");
}

int main() {
    int n, i;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process proc[n];

    for (i = 0; i < n; i++) {
        printf("Enter burst time for process P%d: ", i);
        scanf("%d", &proc[i].burst_time);
        printf("Enter arrival time for process P%d: ", i);
        scanf("%d", &proc[i].arrival_time);
        proc[i].pid = i;
    }

    srtfScheduling(proc, n);
    calculateWaitingTime(proc, n);
    calculateTurnaroundTime(proc, n);

    return 0;
}
```



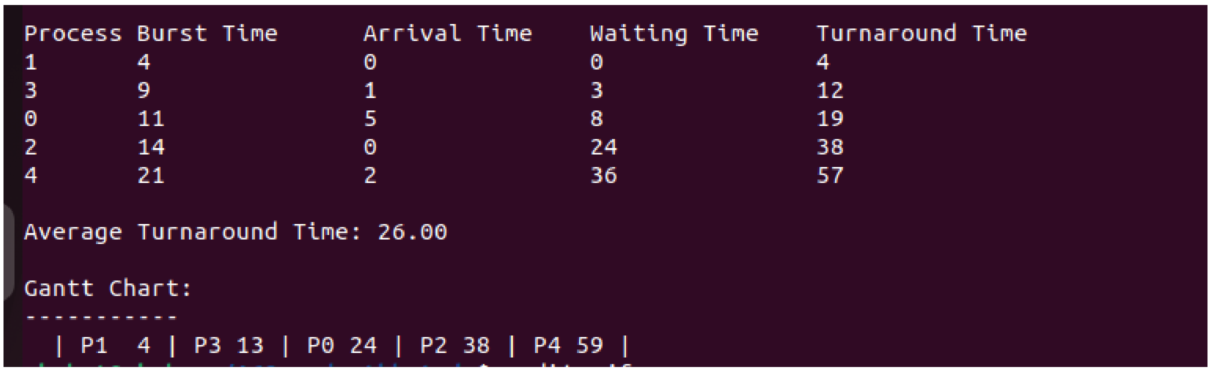
```
akshat@aksha:~/A69_vedantbhutada$ gcc srtf.c
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter the number of processes: 3
Enter burst time for process P0: 9
Enter arrival time for process P0: 0
Enter burst time for process P1: 4
Enter arrival time for process P1: 1
Enter burst time for process P2: 9
Enter arrival time for process P2: 2

Gantt Chart:
-----------
  P0 | P1 | P0 | P2 |

Process Burst Time      Arrival Time    Waiting Time
0       9               0               4
1       4               1               0
2       9               2               11

Average Waiting Time: 5.00

Process Burst Time      Arrival Time    Waiting Time    Turnaround Time
0       9               0               4               13
1       4               1               0               4
2       9               2               11              20

Average Turnaround Time: 12.33
```

b) Non-preemptive SJF

```
#include <stdio.h>

struct Process {
    int pid;
    int burst_time;
    int arrival_time;
    int waiting_time;
    int turnaround_time;
};

void calculateWaitingTime(struct Process proc[], int n) {
    int i, j;
    int total_wt = 0;
    int current_time = 0;

    for (i = 0; i < n; i++) {
        if (current_time < proc[i].arrival_time)
            current_time = proc[i].arrival_time;

        proc[i].waiting_time = current_time - proc[i].arrival_time;
        total_wt += proc[i].waiting_time;
        current_time += proc[i].burst_time;
    }
    float avg_wt = (float)total_wt / n;

    printf("\nProcess\tBurst Time\tArrival Time\tWaiting Time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].arrival_time,
proc[i].waiting_time);

    printf("\nAverage Waiting Time: %.2f\n", avg_wt);
}

void calculateTurnaroundTime(struct Process proc[], int n) {
    int i;
    int total_tat = 0;

    for (i = 0; i < n; i++) {
        proc[i].turnaround_time = proc[i].burst_time + proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
    }

    float avg_tat = (float)total_tat / n;

    printf("\nProcess\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time,
proc[i].arrival_time, proc[i].waiting_time, proc[i].turnaround_time);

    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
}

void sjfScheduling(struct Process proc[], int n) {
    int i, j;
    struct Process temp;

    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (proc[i].burst_time > proc[j].burst_time) {
                temp = proc[i];
                proc[i] = proc[j];
                proc[j] = temp;
            }
        }
    }

    calculateWaitingTime(proc, n);
    calculateTurnaroundTime(proc, n);

    printf("\nGantt Chart:\n");
    printf("-----------\n");
    printf("  ");
    int current_time = proc[0].arrival_time;
    for (i = 0; i < n; i++) {
        printf("| P%d ", proc[i].pid);
        current_time += proc[i].burst_time;
        printf("%2d ", current_time);
    }
    printf("|\n");
}

int main() {
    int n, i;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process proc[n];

    for (i = 0; i < n; i++) {
        printf("Enter burst time for process P%d: ", i);
        scanf("%d", &proc[i].burst_time);
        printf("Enter arrival time for process P%d: ", i);
        scanf("%d", &proc[i].arrival_time);
        proc[i].pid = i;
    }

    sjfScheduling(proc, n);

    return 0;
}
```



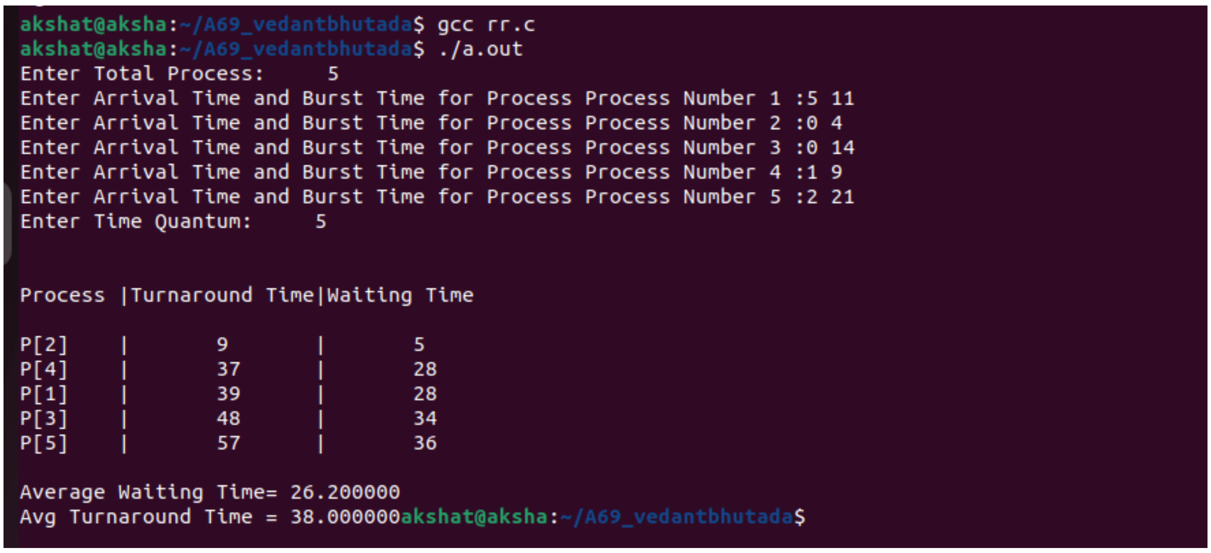
```
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter the number of processes: 5
Enter burst time for process P0: 11
Enter arrival time for process P0: 5
Enter burst time for process P1: 4
Enter arrival time for process P1: 0
Enter burst time for process P2: 14
Enter arrival time for process P2: 0
Enter burst time for process P3: 9
Enter arrival time for process P3: 1
Enter burst time for process P4: 21
Enter arrival time for process P4: 2

Process Burst Time      Arrival Time    Waiting Time
1       4               0               0
3       9               1               3
0       11              5               8
2       14              0               24
4       21              2               36

Average Waiting Time: 14.20
```



```
Process Burst Time      Arrival Time    Waiting Time    Turnaround Time
1       4               0               0               4
3       9               1               3               12
0       11              5               8               19
2       14              0               24              38
4       21              2               36              57

Average Turnaround Time: 26.00

Gantt Chart:
-----------
  | P1  4 | P3 13 | P0 24 | P2 38 | P4 59 |
```

c) ROUND ROBIN

```
#include<stdio.h>

int main()
{

  int cnt,j,n,t,remain,flag=0,tq;
  int wt=0,tat=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(cnt=0;cnt<n;cnt++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",cnt+1);
    scanf("%d",&at[cnt]);
    scanf("%d",&bt[cnt]);
    rt[cnt]=bt[cnt];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&tq);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(t=0,cnt=0;remain!=0;)
  {
    if(rt[cnt]<=tq && rt[cnt]>0)
    {
      t+=rt[cnt];
      rt[cnt]=0;
      flag=1;
    }
    else if(rt[cnt]>0)
    {
      rt[cnt]-=tq;
      t+=tq;
    }
    if(rt[cnt]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",cnt+1,t-at[cnt],t-at[cnt]-bt[cnt]);
      wt+=t-at[cnt]-bt[cnt];
      tat+=t-at[cnt];
      flag=0;
    }
    if(cnt==n-1)
      cnt=0;
    else if(at[cnt+1]<=t)
      cnt++;
    else
      cnt=0;
  }
  printf("\nAverage Waiting Time= %f\n",wt*1.0/n);
  printf("Avg Turnaround Time = %f",tat*1.0/n);

  return 0;
}
```



```
akshat@aksha:~/A69_vedantbhutada$ gcc rr.c
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter Total Process:     5
Enter Arrival Time and Burst Time for Process Process Number 1 :5 11
Enter Arrival Time and Burst Time for Process Process Number 2 :0 4
Enter Arrival Time and Burst Time for Process Process Number 3 :0 14
Enter Arrival Time and Burst Time for Process Process Number 4 :1 9
Enter Arrival Time and Burst Time for Process Process Number 5 :2 21
Enter Time Quantum:     5


Process |Turnaround Time|Waiting Time

P[2]    |       9       |       5
P[4]    |       37      |       28
P[1]    |       39      |       28
P[3]    |       48      |       34
P[5]    |       57      |       36

Average Waiting Time= 26.200000
Avg Turnaround Time = 38.000000akshat@aksha:~/A69_vedantbhutada$
```

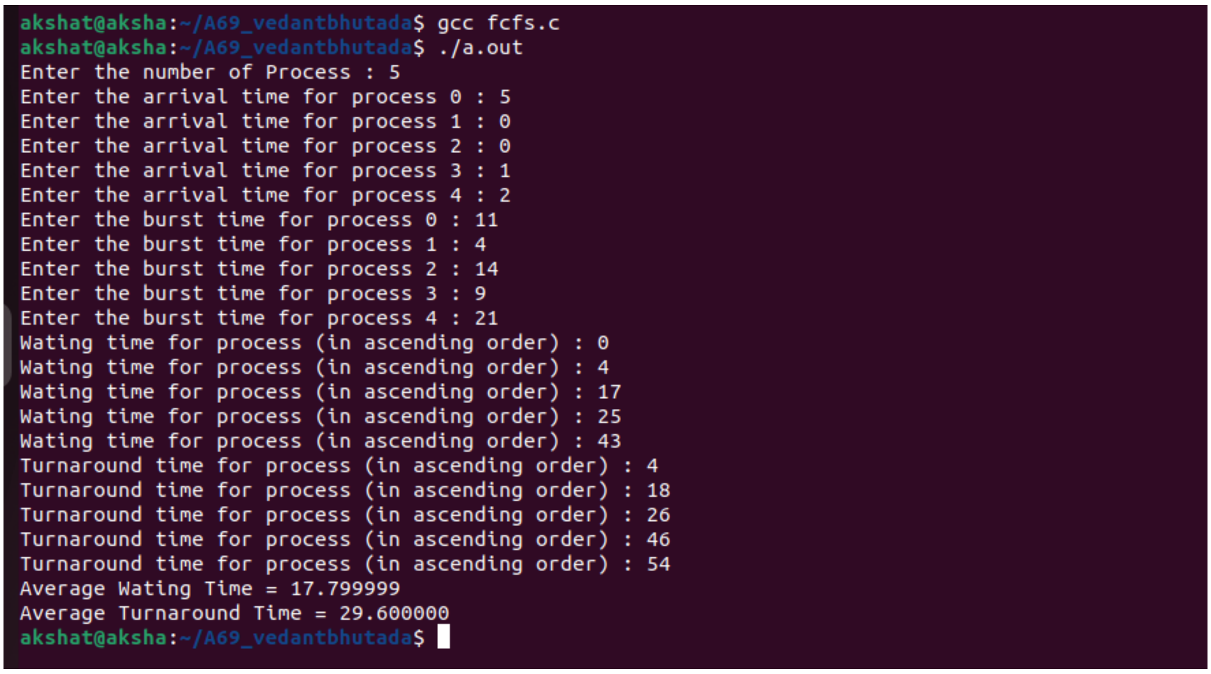
---

d) FCFS

```
#include <stdio.h>
struct process
{
int at;
int bt;
int wt;
int tt;
};
void swap(struct process *A, struct process *B){
struct process temp = *A;
```

```
*A = *B;
*B = temp;
}
int main(){
int num;
printf("Enter the number of Process : ");
scanf("%d", &num);
struct process processes[num];
float avg_wt=0, avg_tt=0;
for (int i=0;i<num;i++){
printf("Enter the arrival time for process %d : ", i);
scanf("%d", &processes[i].at);
}
for (int i=0;i<num;i++){
printf("Enter the burst time for process %d : ", i);
scanf("%d", &processes[i].bt);
}
int i, j;
for (i = 0; i < num - 1; i++){
for (j = 0; j < num - i - 1; j++){
if (processes[j].at > processes[j+1].at){
swap(&processes[j], &processes[j + 1]);
}
}
}
int sum = 0;
processes[0].wt = processes[0].at;
//printf("%d\n", processes[0].wt);
for (int i=0;i<num;i++){
if(i==(num-1)){
break;
}
else{
sum = sum + processes[i].bt;
processes[i+1].wt = sum;
processes[i+1].wt = processes[i+1].wt - processes[i+1].at;
}
}
for (int i=0;i<num;i++){
processes[i].tt = processes[i].bt + processes[i].wt;
}
for (int i=0;i<num;i++){
printf("Wating time for process (in ascending order) : %d \n", processes[i].wt);
}
for (int i=0;i<num;i++){
printf("Turnaround time for process (in ascending order) : %d \n", processes[i].tt);
}
for (int i=0;i<num;i++){
avg_wt = avg_wt + processes[i].wt;
avg_tt = avg_tt + processes[i].tt;
}
```

```
printf("Average Wating Time = %f \n", (avg_wt/num));
printf("Average Turnaround Time = %f \n", (avg_tt/num));
return 0;
}
```



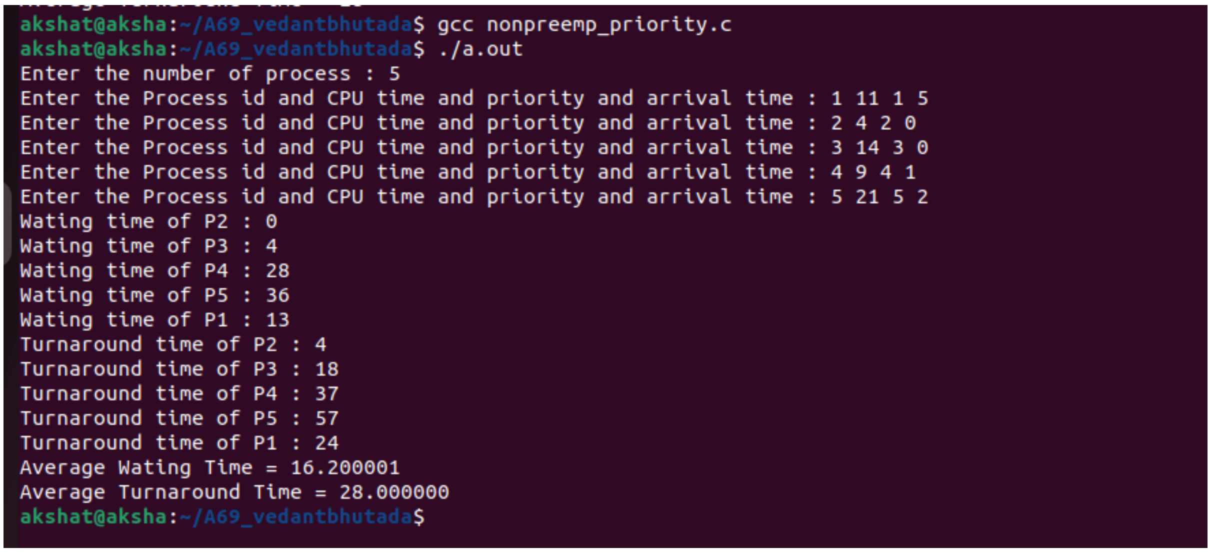
```
akshat@aksha:~/A69_vedantbhutada$ gcc fcfs.c
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter the number of Process : 5
Enter the arrival time for process 0 : 5
Enter the arrival time for process 1 : 0
Enter the arrival time for process 2 : 0
Enter the arrival time for process 3 : 1
Enter the arrival time for process 4 : 2
Enter the burst time for process 0 : 11
Enter the burst time for process 1 : 4
Enter the burst time for process 2 : 14
Enter the burst time for process 3 : 9
Enter the burst time for process 4 : 21
Wating time for process (in ascending order) : 0
Wating time for process (in ascending order) : 4
Wating time for process (in ascending order) : 17
Wating time for process (in ascending order) : 25
Wating time for process (in ascending order) : 43
Turnaround time for process (in ascending order) : 4
Turnaround time for process (in ascending order) : 18
Turnaround time for process (in ascending order) : 26
Turnaround time for process (in ascending order) : 46
Turnaround time for process (in ascending order) : 54
Average Wating Time = 17.799999
Average Turnaround Time = 29.600000
akshat@aksha:~/A69_vedantbhutada$
```

---

e) NON-PREEMPTIVE PRIORITY

```
#include<stdio.h>
struct process {
int id;
int bt;
int at;
int pat;
int pt;
int ppt;
int wt;
int tt;
};
void swap(struct process *A, struct process *B){
struct process temp = *A;
*A = *B;
*B = temp;
}
int main(){
int n;
printf("Enter the number of process : ");
scanf("%d", &n);
struct process p[n];
```

```
float avg_wt=0, avg_tt=0;
int sum=0;
for(int k=0;k<n;k++){
printf("Enter the Process id and CPU time and priority and arrival time : ");
scanf("%d %d %d %d", &p[k].id, &p[k].bt, &p[k].pt, &p[k].at);
p[k].ppt = p[k].pt;
p[k].pat = p[k].at;
sum = sum + p[k].bt + p[k].at;
}
int i, j;
for (i = 0; i < n - 1; i++){
for (j = 0; j < n - i - 1; j++){
if (p[j].at > p[j+1].at){
swap(&p[j], &p[j + 1]);
}
}
}
int count=0;
int min=100;
int index;
for(int k=0;k<n;k++){
min = 100;
for(int l=0;l<n;l++){
if(p[l].ppt<min){
if(p[l].pat<=count){
min = p[l].ppt;
index = l;
}
else{
p[l].ppt = 100;
}
}
}
p[index].wt = count;
count = count + p[index].bt;
p[index].pat = sum;
for(int o=0;o<n;o++){
p[o].ppt = p[o].pt;
}
}
for(int k=0;k<n;k++){
p[k].wt = p[k].wt - p[k].at;
printf("Wating time of P%d : %d \n", p[k].id, p[k].wt);
p[k].tt = p[k].wt + p[k].bt;
}
for (int l=0;l<n;l++){
printf("Turnaround time of P%d : %d \n", p[l].id, p[l].tt);
avg_wt = avg_wt + p[l].wt;
avg_tt = avg_tt + p[l].tt;
}
printf("Average Wating Time = %f \n", (avg_wt/n));
printf("Average Turnaround Time = %f \n", (avg_tt/n));
return 0;
}
```



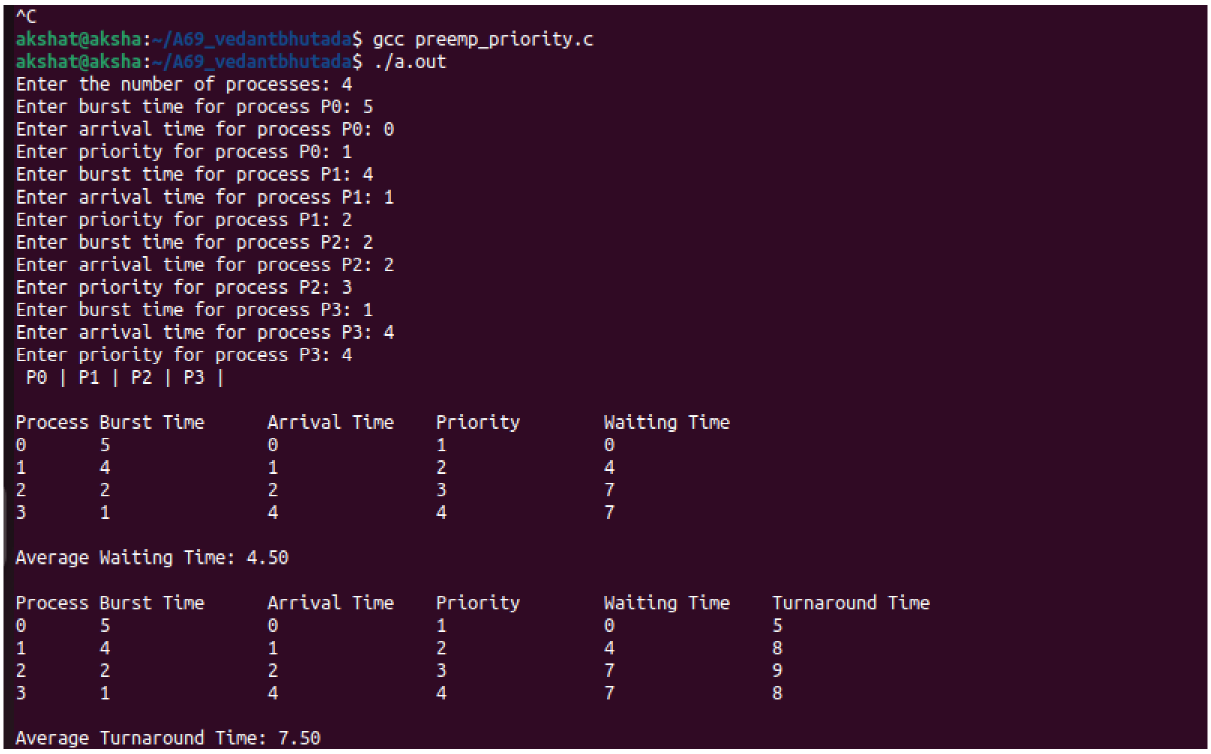
```
akshat@aksha:~/A69_vedantbhutada$ gcc nonpreemp_priority.c
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter the number of process : 5
Enter the Process id and CPU time and priority and arrival time : 1 11 1 5
Enter the Process id and CPU time and priority and arrival time : 2 4 2 0
Enter the Process id and CPU time and priority and arrival time : 3 14 3 0
Enter the Process id and CPU time and priority and arrival time : 4 9 4 1
Enter the Process id and CPU time and priority and arrival time : 5 21 5 2
Wating time of P2 : 0
Wating time of P3 : 4
Wating time of P4 : 28
Wating time of P5 : 36
Wating time of P1 : 13
Turnaround time of P2 : 4
Turnaround time of P3 : 18
Turnaround time of P4 : 37
Turnaround time of P5 : 57
Turnaround time of P1 : 24
Average Wating Time = 16.200001
Average Turnaround Time = 28.000000
akshat@aksha:~/A69_vedantbhutada$
```

---

f) PREEMPTIVE PRIORITY

```
#include <stdio.h>

struct Process {
   int pid;
   int burst_time;
   int arrival_time;
   int remaining_time;
   int priority;
   int waiting_time;
   int turnaround_time;
};

void calculateWaitingTime(struct Process proc[], int n) {
   int i, j;
   int total_wt = 0;

   for (i = 0; i < n; i++) {
      proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;
      total_wt += proc[i].waiting_time;
   }

   float avg_wt = (float)total_wt / n;

   printf("\nProcess\tBurst Time\tArrival Time\tPriority\tWaiting Time\n");
   for (i = 0; i < n; i++)
      printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].arrival_time,
proc[i].priority, proc[i].waiting_time);
```

```
    printf("\nAverage Waiting Time: %.2f\n", avg_wt);
}

void calculateTurnaroundTime(struct Process proc[], int n) {
    int i;
    int total_tat = 0;

    for (i = 0; i < n; i++) {
        proc[i].turnaround_time = proc[i].burst_time + proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
    }

    float avg_tat = (float)total_tat / n;

    printf("\nProcess\tBurst Time\tArrival Time\tPriority\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].arrival_time,
proc[i].priority, proc[i].waiting_time, proc[i].turnaround_time);

    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
}

void preemptivePriorityScheduling(struct Process proc[], int n) {
    int i, j;
    int current_time = 0;
    int completed = 0;
    int prev_process = -1;

    while (completed != n) {
        int highest_priority = 9999;
        int selected_process = -1;

        for (i = 0; i < n; i++) {
            if (proc[i].arrival_time <= current_time && proc[i].remaining_time > 0 && proc[i].priority <
highest_priority) {
                highest_priority = proc[i].priority;
                selected_process = i;
            }
        }

        if (selected_process != -1) {
            if (prev_process != selected_process) {
                if (prev_process != -1)
                    printf("|");

                printf(" P%d ", proc[selected_process].pid);
                prev_process = selected_process;
            }

            proc[selected_process].remaining_time--;
            current_time++;
            if (proc[selected_process].remaining_time == 0) {
                completed++;
                proc[selected_process].turnaround_time = current_time -
proc[selected_process].arrival_time;
            }
        } else {
            current_time++;
        }
    }

    printf("|\n");
}

int main() {
    int n, i;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process proc[n];

    for (i = 0; i < n; i++) {
        printf("Enter burst time for process P%d: ", i);
        scanf("%d", &proc[i].burst_time);
        printf("Enter arrival time for process P%d: ", i);
        scanf("%d", &proc[i].arrival_time);
        printf("Enter priority for process P%d: ", i);
        scanf("%d", &proc[i].priority);
        proc[i].pid = i;
        proc[i].remaining_time = proc[i].burst_time;
    }

    preemptivePriorityScheduling(proc, n);
    calculateWaitingTime(proc, n);
    calculateTurnaroundTime(proc, n);

    return 0;
}
```



```
^C
akshat@aksha:~/A69_vedantbhutada$ gcc preemp_priority.c
akshat@aksha:~/A69_vedantbhutada$ ./a.out
Enter the number of processes: 4
Enter burst time for process P0: 5
Enter arrival time for process P0: 0
Enter priority for process P0: 1
Enter burst time for process P1: 4
Enter arrival time for process P1: 1
Enter priority for process P1: 2
Enter burst time for process P2: 2
Enter arrival time for process P2: 2
Enter priority for process P2: 3
Enter burst time for process P3: 1
Enter arrival time for process P3: 4
Enter priority for process P3: 4
 P0 | P1 | P2 | P3 |

Process Burst Time      Arrival Time    Priority        Waiting Time
0       5               0               1               0
1       4               1               2               4
2       2               2               3               7
3       1               4               4               7

Average Waiting Time: 4.50

Process Burst Time      Arrival Time    Priority        Waiting Time    Turnaround Time
0       5               0               1               0               5
1       4               1               2               4               8
2       2               2               3               7               9
3       1               4               4               7               8

Average Turnaround Time: 7.50
```

Result: Linux C programs on different CPU scheduling policies has been implemented.