FCFS

```
#include<stdio.h>
int main()
{
  int p[10], at[10], bt[10], ct[10], tat[10], wt[10], i, j, temp=0, n;
  float awt=0, atat=0;
  printf("enter no of process you want:");
  scanf("%d",&n);
  printf("enter %d process:", n);
  for(i=0; i<n; i++)
  {
    scanf("%d", &p[i]);
  }
  printf("enter %d arrival time:", n);
  for(i=0; i<n; i++)
  {
    scanf("%d", &at[i]);
  }
  printf("enter %d burst time:", n);
  for(i=0; i<n; i++)
  {
    scanf("%d", &bt[i]);
  }
  for(i=0; i<n; i++)
  {
    for(j=0; j<(n-i); j++)
    {
      if(at[j]>at[j+1])
      {
        temp=p[j+1];
        p[j+1]=p[j];
        p[j]=temp;
        temp=at[j+1];
        at[j+1]=at[j];
        at[j]=temp;
        temp=bt[j+1];
        bt[j+1]=bt[j];
        bt[j]=temp;
      }
    }
  }
```

```
ct[0] = at[0] + bt[0];
for (i=1; i<n; i++)
{ temp=0;
  if (ct[i-1]<at[i])
  { temp = at[i]-ct[i-1];
  }
  ct[i] = ct[i-1]+bt[i]+temp;
}
printf("\nP\t AT\t BT\t CT\t TAT\t WT");
for (i=0; i<n; i++)
{
  tat[i] = ct[i]-at[i];
  wt[i] = tat[i]-bt[i];
  attat += tat[i];
  awt += wt[i];
}
atat = attat/n;
awt = awt/n;
for (i=0; i<n; i++)
{
  printf("\nP%d\t %d\t %d\t %d\t %d\t %d", p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
}
printf("\naverage tat : %f", atat);
printf("\naverage WT : %f", awt);
return 0;
```

Output:

```
enter no of procell you want : 4
enter 4 procell : 1
                  2
                  3
                  4
enter 4 arrivaltime : 0
                      0
                      0
                      0
enter 4 burst time : 7
                     3
                     4
                     6
```

P	A.T	B.T	C.T	TAT	WT
P_1	0	7	7	7	0
P_2	0	3	10	10	7
P_3	0	4	14	14	10
P_4	0	6	20	20	14

average tat : 12.75

average wt : 7.75

SJF (Non-preemptive)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void swap(int *x, int *y)
{
   int temp = *x;
   *x = *y;
   *y = temp;
}
void sortat(int p[], int at[], int bt[], int n)
{ int i,j;
 for(i=0; i<n; i++)
 {
  for(j=i+1; j<n; j++)
  {
   if(at[i]>at[j])
    {
      swap(&p[i], &p[j]);
      swap(&at[i], &at[j]);
      swap(&bt[i], &bt[j]);
    }
   else if(at[i]=at[j]
    {
      if(bt[i]>bt[j]
          swap(&p[i], &p[j]);
          swap(&at[i], &at[j]);
          swap(&bt[i], &bt[j]);
    }
  }
}
void tatwt(int ct[], int at[], int bt[], int tat[], int wt[], int
 {
   int i;
   for(i=0; i<n; i++)
   { tat[i] = ct[i] - at[i];
     wt[i] = tat[i] - bt[i];
```

```
int main()
{ int *p, *at, *bt, *tat, *wt, *ct, pos, i, j, min, n=1000, n;
  float awt=0, atat=0;
  printf("enter no of process:");
  scanf("%d", &n);
  p=(int*)malloc(n*sizeof(int));
 at=(int*)malloc(n*sizeof(int));
 bt=(int*)malloc(n*sizeof(int));
 ct=(int*)malloc(n*sizeof(int));
 wt=(int*)malloc(n*sizeof(int));
 tat=(int*)malloc(n*sizeof(int));
 printf("enter the process");
 for(i=0; i<n; i++)
 {
   scanf("%d", &p[i]);
 }
   printf("enter the arrival time");
   for(i=0; i<n; i++)
   { scanf("%d", &at[i]);
   }
   printf("enter the burst time");
   for(i=0; i<n; i++)
   {
    scanf("%d", &bt[i]);
   }
  sortat(p, at, bt, n);
  ct[0]=at[0]+bt[0];
 for(i=1; i<n; i++)
 {
   for(j=i; j<n; j++)
   {
     if(at[j]<=ct[j-1])
     {
       if(bt[j]<min)
       min=bt[j]; pos=j; }}}
```

```
        swap(&p[i], &p[pos]);
        swap(&at[i], &at[pos]);
        swap(&bt[i], &bt[pos]);
        min=1000;
        ct[i]=ct[i-1]+bt[i];
    }
    tatwt(ct,at,bt,tat,wt,n);
    printf("\nPtt at \t bt\t ct \t tat \t wt");
    for(i=0;i<n;i++)
    {
        printf("\n%d\t %d\t %d\t %d\t %d\t %d",
        p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
    }
    for(i=0;i<n;i++)
    {
        atat+=tat[i];
        awt+=wt[i];
    }
    atat=atat/n;
    awt=awt/n;
    printf("\navg tat=%.2f and avg wt=%.2f",
        atat, awt);
    return 0;
}
```

output

```
Enter the number of processes: 4
Enter the arrival time and burst time for P1: 0 6
Enter the arrival time and burst time for P2: 3 8
Enter the arrival time and burst time for P3: 9 7
Enter the arrival time and burst time for P4: 16 3
```

ProID	AT	BT	WT	TAT	CT
1	0	6	0	6	6
2	3	8	3	11	14
3	9	7	5	12	21
4	16	3	5	8	24

```
Avg WT = 3.25
Avg TAT = 9.25
```

SJF (Preemptive)

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

typedef struct {
    int process_id;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int completion_time;
    int waiting_time;
    int tat;
} procell;

void calculate_times(procell proc[], int n)
{
    int completed = 0, current_time = 0, shortest = -1;
    int min_rt = INT_MAX;
    while (completed != n)
    {
        for (int i = 0; i < n; i++)
        {
            if (proc[i].arrival_time <= current_time && proc[i].remaining_time > 0 && proc[i].remaining_time < min_rt)
            {
                min_rt = proc[i].remaining_time;
                shortest = i;
            }
        }
        if (shortest == -1)
        {
            current_time++;
            continue;
        }
        proc[shortest].remaining_time = 0;
        {
            completed++;
            min_rt = INT_MAX;
        int finish_time = current_time + 1;
        proc[shortest].completion_time = finish_time;
        proc[shortest].waiting_time = finish_time - proc[shortest].burst_time
```

```
proc[shortest].arrival_time;

if (proc[shortest].waiting_time < 0)
    proc[shortest].waiting_time = 0;

proc[shortest].tat = finish_time - proc[shortest].arrival_time;
}

current_time++;
}
}

void print_process(process proc[], int n)
{
printf("PID\tAT\tBT\tCT\tWT\tTAT\t

for (int i=0; i<n; i++)
{
printf("%d\t%d\t%d\t%d\t%d\t%d\n",
proc[i].process_id, proc[i].arrival_time, proc[i].burst_time,
proc[i].completion_time, proc[i].waiting_time, proc[i].tat);
}
}

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

process proc[n];

for (int i=0; i<n; i++)
{
printf("Enter arrival time and burst time for process %d:", i+1);
scanf("%d %d", &proc[i].arrival_time, &proc[i].burst_time);

proc[i].process_id = i+1;
proc[i].remaining_time = proc[i].burst_time;
proc[i].completion_time = 0;
proc[i].waiting_time = 0;
proc[i].tat = 0;
}

calculate_times(proc, n);
print_processes(proc, n);
```

```
float total_wt = 0, total_tat = 0;
for (int i = 0; i < n; i++)
{
    total_wt += proc[i].wt;
    total_tat += proc[i].tat;
}
printf("Avg WT: %.2f\n", total_wt/n);
printf("Avg TAT: %.2f\n", total_tat/n);
}
```

output:

```
Enter process: 4
Enter arrival time and burst time of a process 1: 0 7
Enter arrival time and burst time of a process 2: 8 3
Enter arrival time and burst time of a process 3: 3 2
Enter arrival time and burst time for process 4: 5 6
```

PID	AT	BT	CT	WT	TAT
1	0	7	9	2	9
2	8	3	12	1	4
3	3	2	5	0	2
4	5	6	18	7	13

Avg WT = 2.50

Avg TAT = 7.00.

```c
#include<stdio.h>
int main()
{
    int  p[10],at[10],bt[10],ct[10],tat[10],wt[10],i,j,temp=0,n;
    float awt=0,atat=0;
    printf("enter no of proccess you want:");
    scanf("%d",&n);
    printf("enter %d process:",n);
    for(i=0;i<n;i++)
    {
    scanf("%d",&p[i]);
    }
    printf("enter %d arrival time:",n);
    for(i=0;i<n;i++)
    {
    scanf("%d",&at[i]);
    }
    printf("enter %d burst time:",n);
    for(i=0;i<n;i++)
    {
    scanf("%d",&bt[i]);
    }
    for(i=0;i<n;i++)
    {
     for(j=0;j<(n-i);j++)
    {
      if(at[j]>at[j+1])
     {
        temp=p[j+1];
        p[j+1]=p[j];
        p[j]=temp;
        temp=at[j+1];
        at[j+1]=at[j];
        at[j]=temp;
        temp=bt[j+1];
        bt[j+1]=bt[j];
        bt[j]=temp;
```

```
        temp=bt[j+1];
        bt[j+1]=bt[j];
        bt[j]=temp;
      }
    }
  }
ct[0]=at[0]+bt[0];

for(i=1;i<n;i++)
{
  temp=0;
 if(ct[i-1]<at[i])
 {
    temp=at[i]-ct[i-1];
 }
 ct[i]=ct[i-1]+bt[i]+temp;
}
printf("\np\t A.T\t B.T\t C.T\t TAT\t WT");
for(i=0;i<n;i++)
{
tat[i]=ct[i]-at[i];
wt[i]=tat[i]-bt[i];
atat+=tat[i];
awt+=wt[i];
}
atat=atat/n;
awt=awt/n;
for(i=0;i<n;i++)
{
  printf("\nP%d\t %d\t %d\t %d \t %d \t %d",p[i],at[i],bt[i],ct[i],tat[i],wt[i]);
}
printf("\naverage turnaround time is %f",atat);

printf("\naverage wating timme is %f",awt);
return 0;
}
```

```
enter no of proccess you want:4
enter 4 process:1
2
3
4
enter 4 arrival time:0
0
0
0
enter 4 burst time:7
3
4
6

p       A.T     B.T     C.T     TAT     WT
P1      0       7       7       7       0
P2      0       3       10      10      7
P3      0       4       14      14      10
P4      0       6       20      20      14
average turnaround time is 12.750000
average wating timme is 7.750000
Process returned 0 (0x0)   execution time : 14.678 s
Press any key to continue.
```

```c
#include <stdio.h>

typedef struct {
    int process_id;
    int arrival_time;
    int burst_time;
    int waiting_time;
    int turnaround_time;
    int completion_time;
} Process;

void sort_by_arrival_time(Process proc[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (proc[j].arrival_time > proc[j + 1].arrival_time) {
                Process temp = proc[j];
                proc[j] = proc[j + 1];
                proc[j + 1] = temp;
            }
        }
    }
}

void calculate_times(Process proc[], int n) {
    int current_time = 0;
    int completed = 0;
    int min_burst_time;
    int shortest;

    while (completed != n) {
        shortest = -1;
        min_burst_time = 1000000; // a large value to ensure th

        for (int i = 0; i < n; i++) {
            if (proc[i].arrival_time <= current_time && proc[i].
                min_burst_time = proc[i].burst_time;
                shortest = i;
```

```
        if (shortest == -1) {
            current_time++;
            continue;
        }

        proc[shortest].completion_time = current_time + proc[shortest].burst_time;
        proc[shortest].turnaround_time = proc[shortest].completion_time - proc[shortest].arrival_time;
        proc[shortest].waiting_time = proc[shortest].turnaround_time - proc[shortest].burst_time;

        current_time += proc[shortest].burst_time;
        completed++;
    }
}

void print_processes(Process proc[], int n) {
    printf("Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\tCompletion Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].process_id, proc[i].arrival_time, proc[i].burst_time, p
    }
}

int main() {
    int n;

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

    Process proc[n];

    for (int i = 0; i < n; i++) {
        printf("Enter the arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &proc[i].arrival_time, &proc[i].burst_time);
        proc[i].process_id = i + 1;
        proc[i].completion_time = 0; // Initialize completion time to 0
    }
```

```c
int main() {
    int n;

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

    Process proc[n];

    for (int i = 0; i < n; i++) {
        printf("Enter the arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &proc[i].arrival_time, &proc[i].burst_time);
        proc[i].process_id = i + 1;
        proc[i].completion_time = 0; // Initialize completion time to 0
    }

    sort_by_arrival_time(proc, n);
    calculate_times(proc, n);

    print_processes(proc, n);

    float total_waiting_time = 0, total_turnaround_time = 0;
    for (int i = 0; i < n; i++) {
        total_waiting_time += proc[i].waiting_time;
        total_turnaround_time += proc[i].turnaround_time;
    }

    printf("\nAverage Waiting Time = %.2f\n", total_waiting_time / n);
    printf("Average Turnaround Time = %.2f\n", total_turnaround_time / n);

    return 0;
}
```

```
Enter the number of processes: 4
Enter the arrival time and burst time for process 1: 0
6
Enter the arrival time and burst time for process 2: 3
8
Enter the arrival time and burst time for process 3: 9
7
Enter the arrival time and burst time for process 4: 16
3
```

Process ID	Arrival Time	Burst Time	Waiting Time	Turnaround Time	Completion Time
1	0	6	0	6	6
2	3	8	3	11	14
3	9	7	5	12	21
4	16	3	5	8	24

```
Average Waiting Time = 3.25
Average Turnaround Time = 9.25
```

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

typedef struct {
    int process_id;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int completion_time;
    int waiting_time;
    int turnaround_time;
} Process;

void calculate_times(Process proc[], int n) {
    int completed = 0, current_time = 0, shortest = -1;
    int min_remaining_time = INT_MAX;

    while (completed != n) {
        for (int i = 0; i < n; i++) {
            if (proc[i].arrival_time <= current_time && proc[i].remaining_time > 0 && proc[i].remaining_time < min_remaining_time) {
                min_remaining_time = proc[i].remaining_time;
                shortest = i;
            }
        }

        if (shortest == -1) {
            current_time++;
            continue;
        }

        proc[shortest].remaining_time--;

        if (proc[shortest].remaining_time == 0) {
            completed++;
            min_remaining_time = INT_MAX;

            int finish_time = current_time + 1;
```

```
        if (proc[shortest].waiting_time < 0)
            proc[shortest].waiting_time = 0;

        proc[shortest].turnaround_time = finish_time - proc[shortest].arrival_time;


    current_time++;
}


void print_processes(Process proc[], int n) {
    printf("Process ID\tArrival Time\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].process_id, proc[i].arrival_time, proc[i].burst_time, proc[i].completion_time, proc[i].waiting_time, proc[i].turnaround_time);
    }
}

int main() {
    int n;

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

    Process proc[n];

    for (int i = 0; i < n; i++) {
        printf("Enter the arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &proc[i].arrival_time, &proc[i].burst_time);
        proc[i].process_id = i + 1;
        proc[i].remaining_time = proc[i].burst_time;
        proc[i].completion_time = 0;
        proc[i].waiting_time = 0;
        proc[i].turnaround_time = 0;
```

```c
int main() {
    int n;

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

    Process proc[n];

    for (int i = 0; i < n; i++) {
        printf("Enter the arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &proc[i].arrival_time, &proc[i].burst_time);
        proc[i].process_id = i + 1;
        proc[i].remaining_time = proc[i].burst_time;
        proc[i].completion_time = 0;
        proc[i].waiting_time = 0;
        proc[i].turnaround_time = 0;
    }

    calculate_times(proc, n);
    print_processes(proc, n);

    float total_waiting_time = 0, total_turnaround_time = 0;
    for (int i = 0; i < n; i++) {
        total_waiting_time += proc[i].waiting_time;
        total_turnaround_time += proc[i].turnaround_time;
    }

    printf("\nAverage Waiting Time = %.2f\n", total_waiting_time / n);
    printf("Average Turnaround Time = %.2f\n", total_turnaround_time / n);

    return 0;
}
```

```
Enter the number of processes: 4
Enter the arrival time and burst time for process 1: 0
7
Enter the arrival time and burst time for process 2: 8
3
Enter the arrival time and burst time for process 3: 3
2
Enter the arrival time and burst time for process 4: 5
6
```

Process ID	Arrival Time	Burst Time	Completion Time	Waiting Time	Turnaround Time
1	0	7	9	2	9
2	8	3	12	1	4
3	3	2	5	0	2
4	5	6	18	7	13

```
Average Waiting Time = 2.50
Average Turnaround Time = 7.00
```