Implementation of FCFS Algorithm

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


//Structure to Store AT,BT,WT,TAT
struct process
{
char name[10];
int arr;
int burst;
int pwt;
int ptt;
int ct;
}p[10],temp;


//Structure to Store Gantt Chart
struct chart
{
char cname[10];
int start;
int stop;
int idle;
}c[10];
//Function to Sort the Entered Process
void sort(int n)
  {
  int c,d;
    for(c=0;c<n;c++)
    {
      for(d=0;d<n-c-1;d++)
      {
        if(p[d].arr>p[d+1].arr)
        {
          temp=p[d];
          p[d]=p[d+1];
          p[d+1]=temp;
        }
      }
    }
  }


//Main Program
void main()
{
  int n,i,j,k,l,x,y;
  float wt=0;
  float turn=0;
  printf("Enter the number of process: ");
  scanf("%d",&n);
  __fpurge(stdin);
  for(i=0;i<n;i++)
  {
    printf("\n Process %d",i+1);
    printf("\nName of process: ");
    scanf("%s",&p[i].name);
    printf("Arrival time of process: ");
    scanf("%d",&p[i].arr);
    printf("Burst time of process: ");
    scanf("%d",&p[i].burst);
  }
  sort(n);
  i=0;
  j=0;
  k=0;

//Calculation of Idle Time
  while(i<n)
  {
    if(p[i].arr>k)
    {
      c[j].cname[0]='I';
      c[j].cname[1]='D';
      c[j].cname[2]='L';
      c[j].cname[3]='E';

      c[j].start=k;
      k=p[i].arr;
      c[j].stop=k;
      c[j].idle=c[j].stop-c[j].start;

      printf("The idle time is: %d",c[j].idle);
      j++;
    }
    else
    {
      strcpy(c[j].cname,p[i].name);
      p[i].pwt=k-p[i].arr;
      p[i].ptt=p[i].pwt+p[i].burst;
      p[i].ct=p[i].arr+p[i].burst+p[i].pwt;
      c[j].start=k;
      k=k+p[i].burst;
      c[j].stop=k;
      i++;
      j++;
    }
  }

//Gantt Chart Printing
    printf("\nGantt Chart\n");
    for(i=0;i<j;i++)
    {

      printf("————————————————————————————————————————————————

      printf("\n");
      printf("|");

        for(i=0;i<j;i++)


        printf("\t%s\t|",c[i].cname);
        printf("\n");


    }

    for(i=0;i<j;i++)
    {

      printf("————————————————————————————————————————————————

      printf("\n0");

      for(i=0;i<j;i++)

      printf("\t\t%d",c[i].stop);
      printf("\n");

    }

//Process Table Display
    printf("\n\nProcess Table\n");
    printf("Process Arrival time  Burst time  Turn around time
         Waiting time    Completion time\n");
    i=0;

    while(i<n)
    {
```

```
    printf("%s\t\t%d\t\t%d\t\t%d\t\t%d\t%d",p[i].name,p[i].arr,p[i].burst,p
    printf("\n");
    i++;
    }
    l=0;
    while(l<n)
    {
      wt=wt+p[l].pwt;
      turn=turn+p[l].ptt;
      l++;
    }
    wt=wt/n;
    turn=turn/n;
    printf("Average waiting time is: %2f",wt);
    printf("\n");
    printf(" Average turn around time is: %2f",turn);
    printf("\n");



}
  //End of Program
```

Output

```
Enter the number of process: 4

 Process 1
Name of process: P1
Arrival time of process: 0
Burst time of process: 3

 Process 2
Name of process: P2
Arrival time of process: 5
Burst time of process: 3

 Process 3
Name of process: P4
Arrival time of process: 6
Burst time of process: 6

 Process 4
Name of process: P3
Arrival time of process: 7
Burst time of process: 6
The idle time is: 2
```

```
Gantt Chart
----------------------------------------------------------------------------
| P1   | IDLE   | P2   | P4   | P3   |
----------------------------------------------------------------------------
0    3    5    8    14     20


Process Table
Process  Arrival time  Burst time  Turn around time   Waiting
   time    Completion time
P1      0    3    3    0 3
P2      5    3    3    0 8
P4      6    6    8    2 14
P3      7    6    13     7 20


Average waiting time is: 2.250000
Average turn around time is: 6.750000
```