```
/*................C Program to Implement FCFS ( First Come First Serve ) CPU SCheduling
Algorithm....................

FCFS - A Non-Preemptive Algorithm

This Program works for same as well as different arrival times

*/

#include<stdio.h>

#include <stdlib.h>

struct process_struct

{

  int pid;

  int at;       //Arrival Time

  int bt;       //CPU Burst time

  int ct,wt,tat,rt,start_time;   // Completion, waiting, turnaround, response time

}ps[100];       //Array of structure to store the info of each process.



int findmax(int a, int b)

{

    return a>b?a:b;

}


int comparatorAT(const void * a, const void *b)

{

   int x =((struct process_struct *)a) -> at;

   int y =((struct process_struct *)b) -> at;

   if(x<y)

     return -1;  // No sorting

   else if( x>=y) // = is for stable sort

    return 1;    // Sort

}


int comparatorPID(const void * a, const void *b)
```

```
{
  int x =((struct process_struct *)a) -> pid;
  int y =((struct process_struct *)b) -> pid;
  if(x<y)
    return -1;  // No sorting
  else if( x>=y)
   return 1;    // Sort
}

int main()
{
   int n;
   printf("Enter total number of processes: ");
   scanf("%d",&n);
   float sum_tat=0,sum_wt=0,sum_rt=0;
   int length_cycle,total_idle_time=0;
   float cpu_utilization;

   for(int i=0;i<n;i++)
   {
      printf("\nEnter Process %d Arrival Time: ",i);
      scanf("%d",&ps[i].at);
      ps[i].pid = i ;
   }

   for(int i=0;i<n;i++)
   {
      printf("\nEnter Process %d Burst Time: ",i);
      scanf("%d",&ps[i].bt);
   }
   //sort
   qsort((void *)ps,n, sizeof(struct process_struct),comparatorAT);

  //calculations
   for(int i=0;i<n;i++)
   {
     ps[i].start_time = (i==0) ? ps[i].at : findmax(ps[i].at, ps[i-1].ct);
     ps[i].ct =  ps[i].start_time + ps[i].bt;
     ps[i].tat = ps[i].ct-ps[i].at;
     ps[i].wt = ps[i].tat-ps[i].bt;
     ps[i].rt=ps[i].wt;

     sum_tat += ps[i].tat;
     sum_wt += ps[i].wt;
     sum_rt += ps[i].rt;
     total_idle_time += (i==0) ? 0 : (ps[i].start_time -  ps[i-1].ct);
   }

   length_cycle = ps[n-1].ct - ps[0].start_time;

   //sort so that process ID in output comes in Original order (just for interactivity)
   qsort((void *)ps,n, sizeof(struct process_struct),comparatorPID);

   //Output
   printf("\nProcess No.\tAT\tCPU Burst Time\tCT\tTAT\tWT\tRT\n");
   for(int i=0;i<n;i++)

printf("%d\t\t%d\t%d\t\t%d\t%d\t%d\t%d\n",ps[i].pid,ps[i].at,ps[i].bt,ps[i].ct,ps[i].tat,ps[i].wt
,ps[i].rt);

   printf("\n");

   cpu_utilization = (float)(length_cycle - total_idle_time)/ length_cycle;
```

```
    printf("\nAverage Turn Around time= %f ",sum_tat/n);
    printf("\nAverage Waiting Time= %f ",sum_wt/n);
    printf("\nAverage Response Time= %f ",sum_rt/n);
    printf("\nThroughput= %f",n/(float)length_cycle);
    printf("\nCPU Utilization(Percentage)= %f",cpu_utilization*100);

    printf("\n");
    return 0;
}
```