Round Robin Algorithm

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

struct process
{
  int status,ct,at,wt,bt,tt,left,st;
  char pname[10];
};

struct process p[20];
struct process d[20];

int queue[20];
int n,num=0,front=-1,rear=-1;
int dq();
float avgWait = 0,avgTA = 0;
void displayg();
void enq(int x);
void roundro(struct process p[],int n,int time);
void sort();



main()
{

int i,time;
  printf("Enter no of processes: ");
  scanf("%d",&n);
__fpurge(stdin);
  for(i=0;i<n;i++)
  {
    printf("Enter pid of process %d\n",i+1);
    scanf("%s",&p[i].pname);
    __fpurge(stdin);
    printf("Enter arrival time of process %d\n",i+1);
    scanf("%d",&p[i].at);
    __fpurge(stdin);
    printf("Enter burst time of process %d\n",i+1);
    scanf("%d",&p[i].bt);
    __fpurge(stdin);
    p[i].status=0;
    p[i].left=p[i].bt;
  }

printf("Enter time slice\n");
```

```
scanf("%d",&time);
roundro(p,n,time);
displayg();
}

void roundro(struct process p[],int n,int time)
{
  int flag,i,j,k,ls;
  flag=0;
  avgWait=0;
  avgTA=0;
  ls=0;
  for(i=0;ls<n;)
  {
    for(j=0;j<n;j++)
      if((p[j].status==0)&&(p[j].at<=i))
      {
        enq(j);
      }
      if(flag==0 && front==-1)
      {
        strcpy(d[num].pname,"idle");
        d[num].st=i;
        i++;
        num++;
        flag=1;
      }
      else if(front!=-1)
      {
        k=dq();
        strcpy(d[num].pname,p[k].pname);
        d[num].st=i;
        if(p[k].left<=time)
        {
          d[num].ct=i+p[k].left;
          i=d[num].ct;
          num++;
          ls++;
          p[k].tt=i-p[k].at;
          p[k].wt=p[k].tt-p[k].bt;
          p[k].ct=d[num-1].ct;
        }
        else
        {
          d[num].ct=i+time;
          i=d[num].ct;
          p[k].left=p[k].left-time;
          num++;
```

```
            for(j=0;j<n;j++)
              if((p[j].status==0)&&(p[j].at<=i))
                {
                enq(j);
                }

              enq(k);
            }
          }
          else
          {
            i++;
          }
     }
}
void enq(int x)
  {
    if(rear==-1)
    {
      front=0;
      rear=0;
    }
    else
    {
      rear++;
    }
    queue[rear]=x;
    p[x].status=1;
  }
int dq()
  {
    int item=-1;
    item=queue[front];
    if(front==-1)
    {
      printf("empty queue");
    }
    else if(front==rear)
    {
      front=-1;
      rear=-1;
    }
    else
      front++;
    return item;
  }

void displayg()
```

```
{
  int i,l;
  float AWT=0,ATT=0;
  printf("-");
  for(i=0;i<num;i++)
    printf("--------");

  printf("\n");

  for(i=0;i<num;i++)
    {
      if(!strcmp(d[i].pname,"idle"))
  printf("| idle ");
      else if(!strcmp(d[i].pname," "))
  printf("       ");
      else
  printf("|  %s  ",d[i].pname);
    }
  printf("|");

  printf("\n");

  for(i=0;i<num;i++)
    printf("--------");
  printf("-");

  printf("\n");

  for(i=0;i<num;i++)
    {
      if(!strcmp(d[i].pname,"idle"))
  printf("%d       ",d[i].st);
      else if(!strcmp(d[i].pname," "))
  printf("       ");
      else
  printf("%d       ",d[i].st);
    }
  printf("%d",d[i-1].ct);

  printf("\n\n");


   printf("\n\n\n");
  printf("\n\t\t\tProcessTable\n\t\t\t----------\n");
  printf("-------------------------------------------------------------------
  printf("\nProcess |       AT       |       BT       |       TAT
          |       WT       |       CT          \n");
    i=0;
```

```
    while(i<n)
      {

        printf("\n");
        printf("%s\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t",p[i].pname,p[i].at,p[i]
        printf("\n");
        i++;
    }
    l=0;
    while(l<n)
      {
      AWT+=p[l].wt;
      ATT+=p[l].tt;
      l++;
      }

    AWT /= (float)n;
    ATT /= (float)n;
  printf("-----------------------------------------------------------------
  printf("\n\n\t Average Waiting Time : %f", AWT);
  printf("\n\n\t Average Turn Around Time : %f\n", ATT);




}
void sort()
{
  struct process temp;

  int b,i;
  b=1;
  while(b)
  {
    b=0;
    for(i=0;i<n-1;i++)
    {
      if(p[i].at > p[i+1].at)
      {
        temp=p[i];
        p[i]=p[i+1];
        p[i+1]=temp;
        b=1;
      }
    }
  }
```

```
}
```

Output

```
42813@user:/mnt/42813/oslab$ gcc rr2.c
42813@user:/mnt/42813/oslab$ ./a.out
Enter no of processes: 5
Enter pid of process 1
P0
Enter arrival time of process 1
0
Enter burst time of process 1
4
Enter pid of process 2
P1
Enter arrival time of process 2
0
Enter burst time of process 2
3
Enter pid of process 3
P2
Enter arrival time of process 3
0
Enter burst time of process 3
2
Enter pid of process 4
P3
Enter arrival time of process 4
1
Enter burst time of process 4
1
Enter pid of process 5
P4
Enter arrival time of process 5
13
Enter burst time of process 5
2
Enter time slice
2
```

P0	P1	P2	P3	P0	P1	idle	P4

0 2 4 6 7 9 10 13 15

ProcessTable

```
---------------------------------------------------------------------------
Process |     AT     |     BT     |     TAT     |     WT     |     CT

P0      0     4     9     5     9

P1      0     3     10     7     10

P2      0     2     6     4     6

P3      1     1     6     5     7

P4      13     2     2     0     15
---------------------------------------------------------------------------

    Average Waiting Time : 4.200000

    Average Turn Around Time : 6.600000
42813@user:/mnt/42813/oslab$
```