```
Script started on 2019-03-03 19:27:34+0530
praveen@praveen
praveen@praveen$ cat FCFS.c
#include<stdio.h>
typedef struct process
{
        int id;
        float at, bt, st, ft, wt, tat, rt;
}Process;
Process p[100];
int N = 0;
void line(int n)
{
   for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
   printf("\n");
}
void input()
{
        printf("No. of processes: ");
        while(N == 0) scanf("%d", &N);
        for(int i = 0; i < N; i++)
        {
                printf("Enter AT and BT of P%d: ", i+1);
                scanf("%f %f", &(p[i].at), &(p[i].bt));
                p[i].id = i + 1;
        }
}
void fcfs()
{
        for(int i = 0; i < N - 1; i++)
                for(int j = 0; j < N - 1; j++)
                        if(p[j].at > p[j+1].at)
                        {
                                Process t = p[j];
                                p[j] = p[j+1];
                                p[j+1] = t;
                        }
        float avgwt, avgtat, avgrt;
        p[0].st = p[0].at;
        p[0].ft = p[0].st + p[0].bt;
        p[0].wt = 0;
        p[0].rt = 0;
        p[0].tat = p[0].ft - p[0].at;
        avgwt = p[0].wt; avgtat = p[0].tat; avgrt = p[0].rt;
        for(int i = 1; i < N; i++)
        {
     p[i].st = (p[i-1].ft > p[i].at) ? p[i-1].ft : p[i].at;
                p[i].ft = p[i].st + p[i].bt;
                p[i].wt = p[i].st - p[i].at;
                p[i].rt = p[i].wt;
```

```
            p[i].tat = p[i].ft - p[i].at;
            avgwt += p[i].wt; avgtat += p[i].tat; avgrt += p[i].rt;
        }
        printf("\nFCFS Scheduling:\nPID \t AT \t BT \t ST \t FT \t WT \t TAT \t RT\n");
        avgwt/=N; avgtat/=N; avgrt/=N;
        for(int i = 0; i < N; i++)
        {
            printf("P %d \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f", p[i].id, p[i].at,
p[i].bt, p[i].st, p[i].ft, p[i].wt, p[i].tat, p[i].rt);
            printf("\n");
        }
        printf("Average: WT = %3.2f TAT = %3.2f RT = %3.2f\n\n", avgwt, avgtat, avgrt);
   printf("Gantt chart:\n");
   line(N);
   for(int i = 0; i <= 2; i++)
   {
      if(i==1) { for(int j = 0; j < N; j++) printf("|    P%d    ", p[j].id); printf("|"); }
      else { for(int j = 0; j < N; j++) printf("|          "); printf("|"); }
      printf("\n");
   }
   line(N);
   for(int i = 0; i < N; i++)
   {
      printf("%.1f     %.1f ",p[i].st,p[i].ft);
   }
   printf("\n");
}
int main()
{
        input();
        fcfs();
}
```

```
praveen@praveen- $ gcc FCFS.c -o FCFS
praveen@praveen- $ ./FCFS
No. of processes: 5
Enter AT and BT of P1: 8 20
Enter AT and BT of P2: 2 10
Enter AT and BT of P3: 10 5
Enter AT and BT of P4: 5 6
Enter AT and BT of P5: 25 30

FCFS Scheduling:
PID      AT       BT       ST       FT       WT       TAT      RT
P 2      2.00     10.00    2.00     12.00    0.00     10.00    0.00
P 4      5.00     6.00     12.00    18.00    7.00     13.00    7.00
P 1      8.00     20.00    18.00    38.00    10.00    30.00    10.00
P 3      10.00    5.00     38.00    43.00    28.00    33.00    28.00
P 5      25.00    30.00    43.00    73.00    18.00    48.00    18.00
```

```
Average: WT = 12.60 TAT = 26.80 RT = 12.60

Gantt chart:
--------------------------------------------------------
|        |        |        |        |        |
|   P2   |   P4   |   P1   |   P3   |   P5   |
|        |        |        |        |        |
--------------------------------------------------------
2.0     12.0 12.0     18.0 18.0     38.0 38.0     43.0 43.0     73.0
praveen@praveen$ ls
praveen@praveen$ cat NPSJF.c
#include<stdio.h>
#include<stdlib.h>
typedef struct process {
    int id;
    int st, ft;
}Process;
Process pe[100];
int pid[10], at[10], bt[10], st[10], ft[10], tat[10], rt[10], wt[10];
int curr, n, p_done[10];
float pt = 0, w = 0, t = 0, r = 0, time = 0;
int total = 0, tot = 0;
typedef struct node {
    int data;
    int bt;
    struct node* next;
}Node;
Node *front = NULL;
Node *rear = NULL;
void line(int n)
{
    for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
    printf("\n");
}
Node* newNode(int d, int p)
{
    Node *temp = (Node*)malloc(sizeof(Node));
    temp->data = d;
    temp->bt = p;
    temp->next = NULL;
    return temp;
}
void dequeue(Node** front)
{
    Node* temp = *front;
    (*front) = (*front)->next;
    free(temp);
}
void enqueue(Node** front, int d, int p)
{
    Node* start = (*front);
    Node* temp = newNode(d, p);
    if(start==NULL)
    {
        *front=temp;
    }
    else if ((*front)->bt > p)
    {
        temp->next = *front;
        (*front) = temp;
    }
    else {
        while (start->next != NULL &&  start->next->bt <p)
        {
            start = start->next;
        }
        temp->next = start->next;
        start->next = temp;
    }
}
void update()
{
    int flag = 0;
    if(wt[curr] == -1)
    {
        wt[curr] = st[curr] - at[curr];
        rt[curr] = st[curr] - at[curr];
        r += rt[curr];
        flag = 1;
    }
    else wt[curr] = wt[curr]+(pt-ft[curr]);
    ft[curr] = time;
    pe[tot].ft = ft[curr];
    tot++;
    printf("%d \t %d \t %d \t",st[curr],ft[curr],wt[curr]);
    if(bt[curr]!=0)
    {
        printf("--\t");
        p_done[curr] = 1;
    }
    else
    {
        tat[curr] = ft[curr] - at[curr];
        printf("%d\t",tat[curr]);
    }
    if(flag==1)
    {
        rt[curr] = st[curr] - at[curr];
        r = r + rt[curr];
        printf("%d\n",rt[curr]);
    }
    else
    {
        printf("--\n");
    }
    if(bt[curr]==0)
    {
        t += tat[curr];
        w += wt[curr];
    }
    else
    {
        enqueue(&front,curr,bt[curr]);
    }
}
void initialise()
{
    for(int i=0;i<n;i++)
    {
        wt[i]=-1;
        p_done[i] = 0;
    }
}
int alldone()
{
    total=0;
    for (int i=0;i<n;i++)
    {
        if(p_done[i]==0)
            return 0;
    }
    return 1;
}
Node* addprocess(int time)
{
    for(int i = 0; i < n; i++)
    {
        if(at[i] == time)
        {
            if(p_done[i] == 0)
            {
                enqueue(&front,i,bt[i]);
                p_done[i] = 1;
            }
        }
    }
    return front;
}
int main()
{
```

```
time = 0;
printf("Enter the number of processes: ");
scanf("%d",&n);
for(int i = 0; i < n; i++)
{
    pid[i] = i + 1;
    printf("Enter AT, BT of P%d: ",pid[i]);
    scanf("%d",&at[i]);
    scanf("%d",&bt[i]);
}
initialise();
printf("PID\t AT\t BT\t ST\t FT\t WT\t TAT\t RT\n");
front = addprocess(time);
if(front != NULL)
{
    curr = front->data;
    dequeue(&front);
}
else curr = -1;
while(!alldone()||(front!=NULL)||curr!=-1)
{
    if(curr != -1)
    {
        pt = time;
        st[curr] = time;
        printf("%d\t %d\t %d\t ",pid[curr],at[curr],bt[curr]);
        pe[tot].id = pid[curr];
        pe[tot].st = st[curr];
        while(bt[curr]>0)
        {

            time++;
            bt[curr]--;
            front = addprocess(time);
        }
        update();
    }
    else
    {
        time++;
        front = addprocess(time);
    }
    if(front != NULL)
    {
        curr = front->data;
        dequeue(&front);
    }
    else curr = -1;
}
w/=n; t/=n; r/=n;
```

```
    printf("Average: WT = %.2f TAT = %.2f RT = %.2f\n", w, t, r);
    printf("Gantt chart:\n");
    line(tot);
    for(int i = 0; i <= 2; i++)
    {
        if(i==1) { for(int j = 0; j < tot; j++) printf("|    P%d    ", pe[j].id); printf("|"); }
        else { for(int j = 0; j < tot; j++) printf("|          "); printf("|"); }
        printf("\n");
    }
    line(tot);
    for(int i = 0; i < tot; i++)
    {
        printf("%d         %d ",pe[i].st,pe[i].ft);
    }
    printf("\n");
}
 praveen@praveen$ gcc NPSJF.c -o NPSJF
 praveen@praveen$ ./NPSJF
Enter the number of processes: 5
Enter AT, BT of P1: 1 8
Enter AT, BT of P2: 0 4
Enter AT, BT of P3: 0 2
Enter AT, BT of P4: 1 10
Enter AT, BT of P5: 25 2
PID     AT      BT      ST      FT      WT      TAT     RT
3       0       2       0       2       0       2       0
2       0       4       2       6       2       6       2
1       1       8       6       14      5       13      5
4       1       10      14      24      13      23      13
5       25      2       25      27      0       2       0
Average: WT = 4.00 TAT = 9.20 RT = 8.00
Gantt chart:
-------------------------------------------------------
|       |       |       |       |       |
|   P3   |   P2   |   P1   |   P4   |   P5   |
|       |       |       |       |       |
-------------------------------------------------------
0       2 2       6 6       14 14       24 25       27
 praveen@praveen$ gcc PSJF cat PSJF.c
#include<stdio.h>
#include<stdlib.h>
typedef struct process {
    int id;
    int st, ft;
}Process;
Process pe[100];
int pid[10], at[10], bt[10], st[10], ft[10], tat[10], rt[10], wt[10];
int curr, n, p_done[10];
float pt = 0, w = 0, t = 0, r = 0, time = 0;
int total = 0, tot = 0;
```

```
typedef struct node {
   int data;
   int bt;
   struct node* next;
}Node;
Node *front = NULL;
Node *rear = NULL;
void line(int n)
{
   for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
   printf("\n");
}
Node* newNode(int d, int p)
{
   Node *temp = (Node*)malloc(sizeof(Node));
   temp->data = d;
   temp->bt = p;
   temp->next = NULL;
   return temp;
}
void dequeue(Node** front)
{
   Node* temp = *front;
   (*front) = (*front)->next;
   free(temp);
}
void enqueue(Node** front, int d, int p)
{
   Node* start = (*front);
   Node* temp = newNode(d, p);
   if(start==NULL)
   {
      *front=temp;
   }
   else if ((*front)->bt > p)
   {
      temp->next = *front;
      (*front) = temp;
   }
   else {
      while (start->next != NULL &&  start->next->bt <p)
      {
         start = start->next;
      }
      temp->next = start->next;
      start->next = temp;
   }
}
void update()
{
   int flag = 0;
   if(wt[curr] == -1)
   {
      wt[curr] = st[curr] - at[curr];
      rt[curr] = st[curr] - at[curr];
      r += rt[curr];
      flag = 1;
   }
   else wt[curr] = wt[curr]+(pt-ft[curr]);
   ft[curr] = time;
   pe[tot].ft = ft[curr];
   tot++;
   printf("%d \t %d \t %d \t",st[curr],ft[curr],wt[curr]);
   if(bt[curr]!=0)
   {
      printf("--\t");
      p_done[curr] = 1;
   }
   else
   {
      tat[curr] = ft[curr] - at[curr];
      printf("%d\t",tat[curr]);
   }
   if(flag==1)
   {
      rt[curr] = st[curr] - at[curr];
      r = r + rt[curr];
      printf("%d\n",rt[curr]);
   }
   else
   {
      printf("--\n");
   }
   if(bt[curr]==0)
   {
      t += tat[curr];
      w += wt[curr];
   }
   else
   {
      enqueue(&front,curr,bt[curr]);
   }
}
void initialise()
{
   for(int i=0;i<n;i++)
   {
      wt[i]=-1;
      p_done[i] = 0;
   }
```

```
}
int alldone()
{
    total=0;
    for (int i=0;i<n;i++)
    {
        if(p_done[i]==0)
            return 0;
    }
    return 1;
}
Node* addprocess(int time)
{
    for(int i = 0; i < n; i++)
    {
        if(at[i] == time)
        {
            if(p_done[i] == 0)
            {
                enqueue(&front,i,bt[i]);
                p_done[i] = 1;
            }
        }
    }
    return front;
}
int main()
{
    time = 0;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    for(int i = 0; i < n; i++)
    {
        pid[i] = i + 1;
        printf("Enter AT, BT of P%d: ",pid[i]);
        scanf("%d",&at[i]);
        scanf("%d",&bt[i]);
    }
    initialise();
    printf("PID\t AT\t BT\t ST\t FT\t WT\t TAT\t RT\n");
    front = addprocess(time);
    if(front != NULL)
    {
        curr = front->data;
        dequeue(&front);
    }
    else curr = -1;
    while(!alldone()||(front!=NULL)||curr!=-1)
    {
        if(curr != -1)
        {
            pt = time;
            st[curr] = time;
            printf("%d\t %d\t %d\t ",pid[curr],at[curr],bt[curr]);
            pe[tot].id = pid[curr];
            pe[tot].st = st[curr];
            while(bt[curr]>0)
            {

                time++;
                bt[curr]--;
                front = addprocess(time);
                if(front != NULL && bt[curr] > bt[front->data])
                    break;
            }
            update();
        }
        else
        {
            time++;
            front = addprocess(time);
        }
        if(front != NULL)
        {
            curr = front->data;
            dequeue(&front);
        }
        else curr = -1;
    }
    w/=n; t/=n; r/=n;
    printf("Average: WT = %.2f TAT = %.2f RT = %.2f\n", w, t, r);
    printf("Gantt chart:\n");
    line(tot);
    for(int i = 0; i <= 2; i++)
    {
        if(i==1) { for(int j = 0; j < tot; j++) printf("|    P%d    ", pe[j].id); printf("|"); }
        else { for(int j = 0; j < tot; j++) printf("|          "); printf("|"); }
        printf("\n");
    }
    line(tot);
    for(int i = 0; i < tot; i++)
    {
        printf("%d         %d ",pe[i].st,pe[i].ft);
    }
    printf("\n");
}
```

```
 praveen@praveen$ gcc PSJF.c -o PSJF
 praveen@praveen$ ./PSJF
Enter the number of processes: 6
Enter AT, BT of P1: 5
```

```
10
Enter AT, BT of P2: 4 2
Enter AT, BT of P3: 0 20
Enter AT, BT of P4: 5 1
Enter AT, BT of P5: 11 10
Enter AT, BT of P6: 2 25
PID     AT      BT      ST      FT      WT      TAT     RT
3       0       20      0       4       0       --      0
2       4       2       4       6       0       2       0
4       5       1       6       7       1       2       1
1       5       10      7       17      2       12      2
5       11      10      17      27      6       16      6
3       0       16      27      43      23      43      --
6       2       25      43      68      41      66      41
Average: WT = 12.17 TAT = 23.50 RT = 16.67
Gantt chart:
-----------------------------------------------------------------------------
|       |       |       |       |       |       |       |
|   P3   |   P2   |   P4   |   P1   |   P5   |   P3   |   P6   |
|       |       |       |       |       |       |       |
-----------------------------------------------------------------------------
0       4 4       6 6       7 7       17 17       27 27       43 43       68
 praveen@praveen$ cat  PP.c
#include<stdio.h>
#include<stdlib.h>
typedef struct process {
   int id;
   int st, ft;
}Process;
Process pe[100];
int pid[10], at[10], bt[10], st[10], ft[10], tat[10], rt[10], wt[10],priority[10];
int curr, n, p_done[10];
float pt = 0, w = 0, t = 0, r = 0, time = 0;
int total = 0, tot = 0;
typedef struct node {
   int data;
   int bt;
   struct node* next;
}Node;
Node *front = NULL;
Node *rear = NULL;
void line(int n)
{
   for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
   printf("\n");
}
Node* newNode(int d, int p)
{
   Node *temp = (Node*)malloc(sizeof(Node));
   temp->data = d;
```

```
    temp->bt = p;
    temp->next = NULL;
    return temp;
}
void dequeue(Node** front)
{
    Node* temp = *front;
    (*front) = (*front)->next;
    free(temp);
}
void enqueue(Node** front, int d, int p)
{
    Node* start = (*front);
    Node* temp = newNode(d, p);
    if(start==NULL)
    {
        *front=temp;
    }
    else if ((*front)->bt > p)
    {
        temp->next = *front;
        (*front) = temp;
    }
    else {
        while (start->next != NULL && priority[start->next->data] <p)
        {
            start = start->next;
        }
        temp->next = start->next;
        start->next = temp;
    }
}
void update()
{
    int flag = 0;
    if(wt[curr] == -1)
    {
        wt[curr] = st[curr] - at[curr];
        rt[curr] = st[curr] - at[curr];
        r += rt[curr];
        flag = 1;
    }
    else wt[curr] = wt[curr]+(pt-ft[curr]);
    ft[curr] = time;
    pe[tot].ft = ft[curr];
    tot++;
    printf("%d \t %d \t %d \t",st[curr],ft[curr],wt[curr]);
    if(bt[curr]!=0)
    {
        printf("--\t");
        p_done[curr] = 1;
    }
    else
    {
        tat[curr] = ft[curr] - at[curr];
        printf("%d\t",tat[curr]);
    }
    if(flag==1)
    {
        rt[curr] = st[curr] - at[curr];
        r = r + rt[curr];
        printf("%d\n",rt[curr]);
    }
    else
    {
        printf("--\n");
    }
    if(bt[curr]==0)
    {
        t += tat[curr];
        w += wt[curr];
    }
    else
    {
        enqueue(&front,curr,bt[curr]);
    }
}
void initialise()
{
    for(int i=0;i<n;i++)
    {
        wt[i]=-1;
        p_done[i] = 0;
    }
}
int alldone()
{
    total=0;
    for (int i=0;i<n;i++)
    {
        if(p_done[i]==0)
            return 0;
    }
    return 1;
}
Node* addprocess(int time)
{
    for(int i = 0; i < n; i++)
    {
        if(at[i] == time)
        {
            if(p_done[i] == 0)
            {
                enqueue(&front,i,priority[i]);
                p_done[i] = 1;
            }
        }
    }
    return front;
}
int main()
{
    time = 0;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    for(int i = 0; i < n; i++)
    {
        pid[i] = i + 1;
        printf("Enter details of process %d ",pid[i]);
        printf("\nEnter arrival time: ");
        scanf("%d",&at[i]);
        printf("Enter burst time: ");
        scanf("%d",&bt[i]);
        printf("Enter priority: ");
        scanf("%d",&priority[i]);

    }
    initialise();
    printf("PID\t AT\t BT\tP\t ST\t FT\t WT\t TAT\t RT\n");
    front = addprocess(time);
    if(front != NULL)
    {
        curr = front->data;
        dequeue(&front);
    }
    else curr = -1;
    while(!alldone()||(front!=NULL)||curr!=-1)
    {
        if(curr != -1)
        {
            pt = time;
            st[curr] = time;
            printf("%d\t %d\t %d\t %d\t",pid[curr],at[curr],bt[curr],priority[curr]);
            pe[tot].id = pid[curr];
            pe[tot].st = st[curr];
            while(bt[curr]>0)
            {

                time++;
                bt[curr]--;
                front = addprocess(time);
                if(front != NULL && priority[curr] > priority[front->data])
                    break;
            }
            update();
        }
        else
        {
            time++;
            front = addprocess(time);
        }
        if(front != NULL)
        {
            curr = front->data;
            dequeue(&front);
        }
        else curr = -1;
    }
    w/=n; t/=n; r/=n;
    printf("Average: WT = %.2f TAT = %.2f RT = %.2f\n", w, t, r);
    printf("Gantt chart:\n");
    line(tot);
    for(int i = 0; i <= 2; i++)
    {
        if(i==1) { for(int j = 0; j < tot; j++) printf("|    P%d    ", pe[j].id); printf("|"); }
        else { for(int j = 0; j < tot; j++) printf("|          "); printf("|"); }
        printf("\n");
    }
    line(tot);
    for(int i = 0; i < tot; i++)
    {
        printf("%d         %d ",pe[i].st,pe[i].ft);
    }
    printf("\n");
}
 praveen@praveen$ gcc PP.c -o PP
 praveen@praveen$ ./PP
Enter the number of processes: 5
Enter details of process 1
Enter arrival time: 4
Enter burst time: 6
Enter priority: 8
Enter details of process 2
Enter arrival time: 1
Enter burst time: 20
Enter priority: 6
Enter details of process 3
Enter arrival time: 0
Enter burst time: 3
Enter priority: 7
Enter details of process 4
Enter arrival time: 10
Enter burst time: 7
Enter priority: 1
Enter details of process 5
Enter arrival time: 0
Enter burst time: 7
Enter priority: 20
PID     AT      BT     P        ST      FT      WT      TAT     RT
3       0       3       7       0       1       0       --      0
3       0       2       7       1       2       0       --      --
3       0       1       7       2       3       0       3       --
2       1       20      6       3       10      2       --      2
4       10      7       1       10      17      0       7       0
1       4       6       8       17      18      13      --      13
1       4       5       8       18      19      13      --      --
1       4       4       8       19      20      13      --      --
1       4       3       8       20      21      13      --      --
1       4       2       8       21      22      13      --      --
1       4       1       8       22      23      13      19      --
2       1       13      6       23      36      15      35      --
5       0       7       20      36      43      36      43      36
Average: WT = 12.80 TAT = 21.40 RT = 20.40
Gantt chart:
--------------------------------------------------------------------------------------------------------------------------------------
------------------------
|       |       |       |       |       |       |       |       |       |       |       |       |       |
|   P3   |   P3   |   P3   |   P2   |   P4   |   P1   |   P1   |   P1   |   P1   |   P1   |   P1   |   P2   |
P5   |
|       |       |       |       |       |       |       |       |       |       |       |       |       |
--------------------------------------------------------------------------------------------------------------------------------------
------------------------
0        1 1        2 2        3 3        10 10        17 17        18 18        19 19        20 20        21 21
22 22        23 23        36 36        43
 praveen@praveen$ cat NPP.c
#include<stdio.h>
typedef struct process
{
   int id, p, v;
   float at, bt, st, ft, wt, tat, rt;
}Process;
Process p[100];
int N = 0;
void line(int n)
{
   for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
   printf("\n");
}
void input()
{
   printf("No. of processes: ");
   while(N == 0) scanf("%d", &N);
   for(int i = 0; i < N; i++)
   {
       printf("Enter AT, BT and P of P%d: ", i+1);
       scanf("%f %f %d", &(p[i].at), &(p[i].bt), &(p[i].p));
       p[i].id = i + 1; p[i].v = 0;
   }
}
void sort(Process p[], int r)
{
   for(int i = 0; i < r - 1; i++)
       for(int j = 0; j < r - 1; j++)
           if(p[j].p > p[j+1].p)
           {
               Process t = p[j];
               p[j] = p[j+1];
               p[j+1] = t;
           }
}
void npp()
{
   for(int i = 0; i < N - 1; i++)
       for(int j = 0; j < N - 1; j++)
           if(p[j].at > p[j+1].at)
           {
               Process t = p[j];
               p[j] = p[j+1];
               p[j+1] = t;
           }
   float avgwt, avgtat, avgrt;
   int c = 0, r = 0, ch = 0, i0 = 0;
   Process q[N], pq[N];
   q[0] = p[0];
   pq[0] = p[0];
   for(int i = 1; i < N; i++) {
       if(p[i].at <= q[0].at && p[i].p < q[0].p)
       {    q[0] = p[i]; i0 = i; }
   }
   p[i0].v = 1;
   q[0].v = 1;
   int t = q[0].at;
   do
   {
       ch = 0;
       if(c == 0) {
           q[0].st = q[0].at;
           q[0].ft = q[0].st + q[0].bt;
           q[0].wt = 0;
           q[0].rt = 0;
           q[0].tat = q[0].bt;
           t += q[0].bt;
           c++;
           ch = 1;
           avgwt = 0; avgrt = 0; avgtat = q[0].tat;
       }
       else if(r > 0) {
           q[c] = pq[0];
           q[c].st = (q[c].at > q[c-1].ft) ? q[c].at : q[c-1].ft;
           q[c].ft = q[c].st + q[c].bt;
           q[c].wt = q[c].st - q[c].at;
           q[c].rt = q[c].wt;
           q[c].tat = q[c].ft - q[c].at;
           t += q[c].bt;
           c++;
           ch = 1;
           avgwt += q[c-1].wt; avgrt += q[c-1].rt; avgtat += q[c-1].tat;
       }
       if(ch == 0) t++;
       if(ch == 1 && c > 0) {
           for(int i = 1; i < r; i++) {
               pq[i-1] = pq[i];
           }
           if(r > 0) r--;
       }
       for(int i = 0; i < N; i++)
       {
           if(p[i].v == 0 && p[i].at <= t) {
               p[i].v = 1;
               pq[r] = p[i];
               r++;
           }
       }
       sort(pq,r);
   }while(c < N);
   printf("\nNPP Scheduling:\nPID \t P \t AT \t BT \t ST \t FT \t WT \t TAT \t RT\n");
   avgwt/=N; avgtat/=N; avgrt/=N;
   for(int i = 0; i < N; i++)
   {
       printf("P %d \t %d \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f \t %.2f", q[i].id, q[i].p, q[i].at,
q[i].bt, q[i].st, q[i].ft, q[i].wt, q[i].tat, q[i].rt);
       printf("\n");
   }
   printf("Average: WT = %3.2f TAT = %3.2f RT = %3.2f\n\n", avgwt, avgtat, avgrt);
   printf("Gantt chart:\n");
   line(N);
   for(int i = 0; i <= 2; i++)
   {
       if(i==1) { for(int j = 0; j < N; j++) printf("|    P%d    ", q[j].id); printf("|"); }
       else { for(int j = 0; j < N; j++) printf("|          "); printf("|"); }
       printf("\n");
   }
   line(N);
   for(int i = 0; i < N; i++)
   {
       printf("%.1f     %.1f ",q[i].st,q[i].ft);
   }
   printf("\n");
}
int main()
{
   input();
   npp();
}

 praveen@praveen$ gcc NPP.c -o NPP
 praveen@praveen$ ./NPP
No. of processes: 7
Enter AT, BT and P of P1: 0 20 6
Enter AT, BT and P of P2: 6 4 8
Enter AT, BT and P of P3: 2 30 4
Enter AT, BT and P of P4: 6 1 9
Enter AT, BT and P of P5: 0 6 5
Enter AT, BT and P of P6: 10 20 1
Enter AT, BT and P of P7: 7 2 2

NPP Scheduling:
PID     P       AT      BT      ST      FT      WT      TAT     RT
P 5     5       0.00    6.00    0.00    6.00    0.00    6.00    0.00
P 3     4       2.00    30.00   6.00    36.00   4.00    34.00   4.00
P 6     1       10.00   20.00   36.00   56.00   26.00   46.00   26.00
P 7     2       7.00    2.00    56.00   58.00   49.00   51.00   49.00
P 1     6       0.00    20.00   58.00   78.00   58.00   78.00   58.00
P 2     8       6.00    4.00    78.00   82.00   72.00   76.00   72.00
P 4     9       6.00    1.00    82.00   83.00   76.00   77.00   76.00
Average: WT = 40.71 TAT = 52.57 RT = 40.71

Gantt chart:
-----------------------------------------------------------------------------
|       |       |       |       |       |       |       |
|   P5   |   P3   |   P6   |   P7   |   P1   |   P2   |   P4   |
|       |       |       |       |       |       |       |
-----------------------------------------------------------------------------
0.0     6.0 6.0     36.0 36.0     56.0 56.0     58.0 58.0     78.0 78.0     82.0 82.0     83.0
 praveen@praveen$ cat RR.c
#include<stdio.h>
typedef struct process
{
   int id;
   float at, bt, st, ft, wt, tat, rt;
```

```
}Process;
Process p[100], pe[100];
int N = 0, ts = 1;
void line(int n)
{
    for(int l = 0; l < n; l++) for(int i = 0; i < 11 + (1/(l+1)); i++) printf("-");
    printf("\n");
}
void input()
{
    printf("No. of processes: ");
    while(N == 0) scanf("%d", &N);
    for(int i = 0; i < N; i++)
    {
        printf("Enter AT and BT of P%d: ", i+1);
        scanf("%f %f", &(p[i].at), &(p[i].bt));
        p[i].id = i + 1;
    }
    printf("Time slice: ");
    scanf("%d", &ts);
}
void rr()
{
    for(int i = 0; i < N - 1; i++)
        for(int j = 0; j < N - 1; j++)
            if(p[j].at > p[j+1].at)
            {
                Process t = p[j];
                p[j] = p[j+1];
                p[j+1] = t;
            }
    float avgwt, avgtat, avgrt;
    int tot = 0, c = N, qn = 1, t = p[0].at, V[100] = {0}, cc = 0;
    Process qu[N];
    qu[0] = p[0];
    pe[0] = p[0];
    while(c != 0) {
        cc = 0;
        for(int i = qn; i < N; i++)
        {
            if(p[i].at <= t) {
                qu[qn] = p[i];
                qn++;
            }
        }
        for(int i = 0; i < qn; i++)
        {
            if(V[i] != -1) {
                pe[tot].id = qu[i].id;
                pe[tot].at = qu[i].at;
            pe[tot].bt = qu[i].bt;
            if(tot == 0) {
                pe[tot].st = qu[i].at;
                pe[tot].ft = (qu[i].bt > ts) ? (qu[i].at + ts) : (qu[i].at + qu[i].bt);
                qu[i].bt -= ts;
                pe[tot].wt = 0;
                pe[tot].rt = 0;
                if(qu[i].bt <= 0) pe[tot].tat = pe[tot].ft - pe[tot].at;
                else pe[tot].tat = -1;
                if(pe[tot].tat != -1) qu[i].tat = pe[tot].tat;
                if(pe[tot].rt != -1) qu[i].rt = pe[tot].rt;
                qu[i].wt = 0;
                qu[i].ft = pe[tot].ft;
                V[i] = 1;
                if(qu[i].bt <= 0){ V[i] = -1; c--;}
            }
            else {
                if(pe[tot].at <= pe[tot-1].ft) pe[tot].st = pe[tot-1].ft;
                else pe[tot].st = pe[tot].at;
                pe[tot].ft = (qu[i].bt > ts) ? (pe[tot].st + ts) : (pe[tot].st + qu[i].bt);
                qu[i].bt -= ts;
                if(V[i] == 1) pe[tot].wt = qu[i].wt + (pe[tot].st - qu[i].ft);
                else pe[tot].wt = pe[tot].st - pe[tot].at;
                if(V[i] == 0) pe[tot].rt = pe[tot].st - pe[tot].at;
                else pe[tot].rt = -1;
                if(qu[i].bt <= 0) pe[tot].tat = pe[tot].ft - pe[tot].at;
                else pe[tot].tat = -1;
                qu[i].ft = pe[tot].ft;
                qu[i].wt = pe[tot].wt;
                if(pe[tot].tat != -1) qu[i].tat = pe[tot].tat;
                if(pe[tot].rt != -1) qu[i].rt = pe[tot].rt;
                V[i] = 1;
                if(qu[i].bt <= 0){ V[i] = -1; c--;}
            }
            tot++; t+= ts; cc = 1;
        }
        for(int i = qn; i < N; i++)
        {
            if(p[i].at <= t) {
                qu[qn] = p[i];
                qn++;
            }
        }
    }
    if(cc == 0) t += 1;
}
printf("\nRR Scheduling:\nPID \t AT \t BT \t ST \t FT \t WT \t TAT \t RT\n");
for(int i = 0; i < tot; i++) {
    if(pe[i].tat != -1) {
        avgtat += pe[i].tat;
```

```
            avgwt += pe[i].wt;
        }
        if(pe[i].rt > 0) avgrt += pe[i].rt;
    }
    avgwt/=N; avgtat/=N; avgrt/=N;
    for(int i = 0; i < tot; i++)
    {
        printf("P %d \t %.2f \t %.2f \t %.2f \t %.2f \t ", pe[i].id, pe[i].at, pe[i].bt, pe[i].st, pe[i].ft);
        if(pe[i].wt < 0)
        {
            printf("-- \t ");
        }
        else {
            printf("%.2f \t ",pe[i].wt);
        }
        if(pe[i].tat < 0)
        {
            printf("-- \t ");
        }
        else {
            printf("%.2f \t ",pe[i].tat);
        }
        if(pe[i].rt < 0)
        {
            printf("-- \t ");
        }
        else {
            printf("%.2f \t ",pe[i].rt);
        }
        printf("\n");
    }
    printf("Average: WT = %3.2f TAT = %3.2f RT = %3.2f\n\n", avgwt, avgtat, avgrt);
    printf("Gantt chart:\n");
    line(tot);
    for(int i = 0; i <= 2; i++)
    {
        if(i==1) { for(int j = 0; j < tot; j++) printf("|    P%d    ", pe[j].id); printf("|"); }
        else { for(int j = 0; j < tot; j++) printf("|          "); printf("|"); }
        printf("\n");
    }
    line(tot);
    for(int i = 0; i < tot; i++)
    {
        printf("%.1f     %.1f ",pe[i].st,pe[i].ft);
    }
    printf("\n");
}
int main()
{
    input();
    rr();
}
 praveen@praveen$ gcc RR.c -o RR
 praveen@praveen$ / / ./RR
No. of processes: 7
Enter AT and BT of P1: 0 10
Enter AT and BT of P2: 4 15
Enter AT and BT of P3: 5 7
Enter AT and BT of P4: 2  30
Enter AT and BT of P5: 14 20
Enter AT and BT of P6: 20 10
Enter AT and BT of P7: 24 2
Time slice: 4

RR Scheduling:
PID     AT      BT      ST      FT      WT      TAT     RT
P 1     0.00    10.00   0.00    4.00    0.00    --      0.00
P 4     2.00    30.00   4.00    8.00    2.00    --      2.00
P 2     4.00    15.00   8.00    12.00   4.00    --      4.00
P 3     5.00    7.00    12.00   16.00   7.00    --      7.00
P 5     14.00   20.00   16.00   20.00   2.00    --      2.00
P 6     20.00   10.00   20.00   24.00   0.00    --      0.00
P 7     24.00   2.00    24.00   26.00   0.00    2.00    0.00
P 1     0.00    6.00    26.00   30.00   22.00   --      --
P 4     2.00    26.00   30.00   34.00   24.00   --      --
P 2     4.00    11.00   34.00   38.00   26.00   --      --
P 3     5.00    3.00    38.00   41.00   29.00   36.00   --
P 5     14.00   16.00   41.00   45.00   23.00   --      --
P 6     20.00   6.00    45.00   49.00   21.00   --      --
P 1     0.00    2.00    49.00   51.00   41.00   51.00   --
P 4     2.00    22.00   51.00   55.00   41.00   --      --
P 2     4.00    7.00    55.00   59.00   43.00   --      --
P 5     14.00   12.00   59.00   63.00   37.00   --      --
P 6     20.00   2.00    63.00   65.00   35.00   45.00   --
P 4     2.00    18.00   65.00   69.00   51.00   --      --
P 2     4.00    3.00    69.00   72.00   53.00   68.00   --
P 5     14.00   8.00    72.00   76.00   46.00   --      --
P 4     2.00    14.00   76.00   80.00   58.00   --      --
P 5     14.00   4.00    80.00   84.00   50.00   70.00   --
P 4     2.00    10.00   84.00   88.00   62.00   --      --
P 4     2.00    6.00    88.00   92.00   62.00   --      --
P 4     2.00    2.00    92.00   94.00   62.00   92.00   --
Average: WT = 38.57 TAT = 52.00 RT = 2.14
```

```
Gantt chart:
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------
|       |       |       |       |       |       |       |       |       |       |       |       |       |       |
|       |       |       |       |       |       |       |       |       |       |       |       |
|   P1   |   P4   |   P2   |   P3   |   P5   |   P6   |   P7   |   P1   |   P4   |   P2   |   P3   |   P5   |
P6   |   P1   |   P4   |   P2   |   P5   |   P6   |   P4   |   P2   |   P5   |   P4   |   P5   |   P4   |   P4
|   P4   |
|       |       |       |       |       |       |       |       |       |       |       |       |       |       |
|       |       |       |       |       |       |       |       |       |       |       |       |
----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------
0.0     4.0 4.0     8.0 8.0     12.0 12.0     16.0 16.0     20.0 20.0     24.0 24.0     26.0 26.0     30.0 30.0
34.0 34.0     38.0 38.0     41.0 41.0     45.0 45.0     49.0 49.0     51.0 51.0     55.0 55.0     59.0 59.0
63.0 63.0     65.0 65.0     69.0 69.0     72.0 72.0     76.0 76.0     80.0 80.0     84.0 84.0     88.0 88.0
92.0 92.0     94.0
 praveen@praveen$  exit


Script done on 2019-03-03 19:43:33+0530
```