Round-Robin Scheduling

Code:

```
#include<stdio.h>

struct Process{
    int id, at, bt, ct, tat, wt, remt;
};

void roundrobin(struct Process p[], int n, int tq){
    int comp=0, curr=0, ttt=0, twt=0;
    int queue[20], visited[n], rear=0, front=0;
    for(int i=0;i<n;i++){
        visited[i]=0;
        p[i].remt=p[i].bt;
    }

    for(int i=0;i<n;i++){
        if(p[i].at==0){
            visited[i]=1;
            queue[rear++]=i;
        }
    }

    while(comp<n){
        if(front==rear){
            curr++;
            for(int i=0;i<n;i++){
                if(p[i].at<=curr && !visited[i]){
                    visited[i]=1;
                    queue[rear++]=i;
                }
            }
            continue;
        }

        int idx=queue[front++];
        int exec_time=(p[idx].remt>tq)?tq:p[idx].remt;
        curr+=exec_time;
        p[idx].remt-=exec_time;

        for(int i=0;i<n;i++){
            if(p[i].at<=curr && !visited[i]){
```

```
            visited[i]=1;
            queue[rear++]=i;
        }
    }

    if(p[idx].remt==0){
        comp++;
        p[idx].ct=curr;
        p[idx].tat=p[idx].ct-p[idx].at;
        p[idx].wt=p[idx].tat-p[idx].bt;
        ttt+=p[idx].tat;
        twt+=p[idx].wt;
    }
    else{
        queue[rear++]=idx;
    }
  }
  printf("Process\tAT\tBT\tCT\tTAT\tWT\n");
  for(int i=0;i<n;i++){
    printf("P%d\t%d\t%d\t%d\t%d\t%d\n",p[i].id, p[i].at, p[i].bt, p[i].ct, p[i].tat, p[i].wt);
  }
  printf("Average TAT: %.2f units\n",(float)ttt/n);
  printf("Average WT: %.2f units\n",(float)twt/n);
}

int main(){
  int n,tq;
  printf("Enter the number of processes:");
  scanf("%d", &n);
  struct Process p[n];
  printf("Enter the arrival and burst time for process:\n");
  for(int i=0;i<n;i++){
    p[i].id=i+1;
    printf("Process %d: ",i+1);
    scanf("%d %d", &p[i].at, &p[i].bt);
  }
  printf("Enter the time quantum:");
  scanf("%d", &tq);
  roundrobin(p,n,tq);
  return 0;
}
```

Output:

```
Enter the number of processes:5
Enter the arrival time and burst time for processes:
Process 1: 0 5
Process 2: 1 3
Process 3: 2 1
Process 4: 3 2
Process 5: 4 3
Enter the time quantum:2
Process AT      BT      CT      TAT     WT
P1      0       5       13      13      8
P2      1       3       12      11      8
P3      2       1       5       3       2
P4      3       2       9       6       4
P5      4       3       14      10      7

Average TAT: 8.60
Average WT: 5.80

Process returned 0 (0x0)   execution time : 34.035 s
Press any key to continue.
```

Priority Scheduling (Preemptive and Non-Preemptive)

Code:

```
//Lower the number, higher the priority.

#include<stdio.h>
#define SIZE 10

struct Process{
    int id, at, bt, priority, ct, tat, wt, remt;
};

void calc_times(struct Process p[], int n, int ispreemptive){
    int time=0, minidx, comp=0, ttt=0, twt=0;
    int executed[SIZE]={0};

    while(comp<n){
        minidx=-1;
        for(int i=0;i<n;i++){
            if(p[i].at<=time && !executed[i]){
                if(minidx==-1 || p[i].priority<p[minidx].priority)
                    minidx=i;
```

```
            }
        }

        if(minidx==-1){
            time++;
            continue;
        }

        if(ispreemptive){
            p[minidx].remt--;
            time++;

            if(p[minidx].remt==0){
                p[minidx].ct=time;
                p[minidx].tat=p[minidx].ct-p[minidx].at;
                p[minidx].wt=p[minidx].tat-p[minidx].bt;
                executed[minidx]=1;
                comp++;
                ttt+=p[minidx].tat;
                twt+=p[minidx].wt;
            }
        }
        else{
            time+=p[minidx].bt;
            p[minidx].ct=time;
            p[minidx].tat=p[minidx].ct-p[minidx].at;
            p[minidx].wt=p[minidx].tat-p[minidx].bt;
            comp++;
            executed[minidx]=1;
            ttt+=p[minidx].tat;
            twt+=p[minidx].wt;
        }
    }
    printf("Process\tAT\tBT\tPriority\tCT\tTAT\tWT\n");
    for(int i=0;i<n;i++)
        printf("P%d\t%d\t%d\t%d\t\t%d\t%d\t%d\n", p[i].id, p[i].at, p[i].bt, p[i].priority, p[i].ct, p[i].tat,
p[i].wt);
    printf("Average TAT: %.2f units\n", (float)ttt/n);
    printf("Average WT: %.2f units\n", (float)twt/n);
}

int main(){
    int n, ispreemptive;
    printf("Enter the number of processes:");
```

```
    scanf("%d", &n);
    struct Process p[n];
    printf("Enter the arrival time, burst time and priority for processes:\n");
    for(int i=0;i<n;i++){
        p[i].id=i+1;
        printf("Process %d: ", i+1);
        scanf("%d %d %d", &p[i].at, &p[i].bt, &p[i].priority);
        p[i].remt=p[i].bt;
    }
    printf("Enter:\n0.Non-Preemptive\n1.Preemptive\n");
    scanf("%d", &ispreemptive);
    calc_times(p, n, ispreemptive);
    return 0;
}
```

Output:

Non-Preemptive

```
Enter the number of processes:5
Enter the arrival time, burst time and priority for processes:
Process 1: 0 3 5
Process 2: 2 2 3
Process 3: 3 5 2
Process 4: 4 4 4
Process 5: 6 1 1
Enter:
0.Non-Preemptive
1.Preemptive
0
Process AT      BT      Priority        CT      TAT     WT
P1      0       3       5               3       3       0
P2      2       2       3               11      9       7
P3      3       5       2               8       5       0
P4      4       4       4               15      11      7
P5      6       1       1               9       3       2
Average TAT: 6.20 units
Average WT: 3.20 units

Process returned 0 (0x0)   execution time : 30.208 s
Press any key to continue.
```

Preemptive

```
Enter the number of processes:5
Enter the arrival time, burst time and priority for processes:
Process 1: 0 3 5
Process 2: 2 2 3
Process 3: 3 5 2
Process 4: 4 4 4
Process 5: 6 1 1
Enter:
0.Non-Preemptive
1.Preemptive
2
Process AT      BT      Priority        CT      TAT     WT
P1      0       3       5               15      15      12
P2      2       2       3               10      8       6
P3      3       5       2               9       6       1
P4      4       4       4               14      10      6
P5      6       1       1               7       1       0
Average TAT: 8.00 units
Average WT: 5.00 units

Process returned 0 (0x0)   execution time : 25.550 s
Press any key to continue.
```