Assignment 3: CPU Scheduling Algorithms

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

struct process {
    int p; //process
    int bt; //burst time (initial)
    int rbt; //burst time (remaining)
    int at; //arrival time
    int wt; //waiting time
    int tat; //turn around time
    int ct; //completion time
    int priority; //priority
};

//function to swap structure objects
void swap_P(struct process * P, int i, int j){
    struct process t = P[i];
    P[i] = P[j];
    P[j] = t;
}

//first come first serve
void FCFS_Algo(int n, struct process P[]){
    if(n <= 0){
        return;
    }
    //sort the processes according to arrival time
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(P[i].at > P[j].at){
                swap_P(P, i, j);
            }
        }
    }
    printf("\n=== First Come First Serve ===\n");
    P[0].wt = 0;
    P[0].ct = P[0].bt;
    P[0].tat = P[0].bt;
    for(int i=1; i<n; i++){
        P[i].wt = P[i-1].bt + P[i-1].wt;
        P[i].tat = P[i].wt + P[i].bt;
        P[i].ct = P[i].tat + P[i].at;
    }
}

//shortest job first
void SJF_Algo(int n, struct process P[], int preemptive){
    if(n <= 0){
        return;
    }
    //sort the processes according to arrival time
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(P[i].at > P[j].at){
                swap_P(P, i, j);
            }
        }
    }
    int time = 0;
        int completed = 0;
        int min, index;
        int done[1000] = {0};
    //shortest remaining time first(SJF preemptive)
    if(preemptive){
        for(int i = 0; i < n; i++){
            P[i].rbt = P[i].bt;
        }
        while(completed < n){
            min = __INT32_MAX__;
            index = -1;
            // Find the process with the shortest remaining time that has
arrived
            for(int i = 0; i < n; i++){
                if(P[i].at <= time && !done[i] && P[i].rbt < min){
                    min = P[i].rbt;
                    index = i;
                }
            }
            if(index == -1){
                time++;
            }
            else{
                P[index].rbt -= 1;
                time++;
                if(P[index].rbt == 0){
                    P[index].ct = time;
                    P[index].tat = P[index].ct - P[index].at;
                    P[index].wt = P[index].tat - P[index].bt;
                    done[index] = 1;
                    completed++;
                }
            }
        }
        printf("\n=== Shortest Remaining Time First (SJF Preemptive) ===\n");
    }
    else{
        while(completed < n){
        min = __INT32_MAX__;
            index = -1;
            //find the shortest job at current time
            for(int i=0; i<n; i++){
                if(P[i].at <= time && done[i] == 0 && P[i].bt < min){
                    min = P[i].bt;
                    index = i;
                }
            }
            if(index == -1){
                // No process is ready to execute, move time forward
                time++;
            }
            else{
                P[index].wt = time - P[index].at;
                P[index].ct = time + P[index].bt;
                P[index].tat = P[index].ct - P[index].at;
                time = P[index].ct;
                done[index] = 1;
                completed++;
            }
        }
        printf("\n=== Shortest Job First ===\n");
    }
}

//priority algorithm
void PS_Algo(int n, struct process P[], int preemptive){
    if(n <= 0){
        return;
    }
    //sort the processes according to arrival time
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(P[i].at > P[j].at){
                swap_P(P, i, j);
            }
        }
    }
    int time = 0;
    int completed = 0;
    int done[1000] = {0};
    int hp, index; //hp is the current highest priority (numerically lowest)

    if(preemptive){
        for(int i = 0; i < n; i++){
            P[i].rbt = P[i].bt;
        }
        while (completed < n) {
            hp = __INT32_MAX__;
            index = -1;
            // Find the process with the highest priority (numerically lowest)
that has arrived
            for(int i=0; i<n; i++){
                if (P[i].at <= time && done[i] == 0 && P[i].rbt > 0 &&
P[i].priority < hp) {
                    hp = P[i].priority;
                    index = i;
                }
            }
            if(index == -1){
                time++;
            }
            else{
                P[index].rbt -= 1;
                time++;
                if (P[index].rbt == 0) {
                    P[index].ct = time;
                    P[index].tat = P[index].ct - P[index].at;
                    P[index].wt = P[index].tat - P[index].bt;
                    done[index] = 1;
                    completed++;
                }
            }
        }
        printf("\n=== Priority Scheduling (Preemptive) ===\n");
    }
    else{
        while(completed < n){
            index = -1;
            hp = __INT32_MAX__;
            //search for unserved process with highest priority from the
arrived processes
            for(int i=0; i<n; i++){
                if(P[i].at <= time && done[i] == 0 && P[i].priority < hp){
                    hp = P[i].priority;
                    index = i;
                }
            }
            if(index == -1){
                time++;
            }
            else{
                P[index].wt = time - P[index].at;
                P[index].ct = time + P[index].bt;
                P[index].tat = P[index].ct - P[index].at;
                time = P[index].ct;
                done[index] = 1;
                completed++;
            }
        }
        printf("\n=== Priority Scheduling (Non Preemptive) ===\n");
    }
}

//round robin
void RR_Algo(int n, struct process P[]){
    if(n <= 0){
        return;
    }
    int quantum;
    printf("Time quantum : ");
    scanf("%d", &quantum);
    int time = 0;
```

```c
    int completed = 0;
    int i;
    for(i=0; i<n; i++){
        P[i].rbt = P[i].bt;
    }

    while (completed < n) {
        int all_done = 1;
        for(i = 0; i < n; i++){
            if (P[i].at <= time && P[i].rbt > 0) {
                all_done = 0;
                int time_reduction = (P[i].rbt < quantum) ? P[i].rbt : quantum;
                P[i].rbt -= time_reduction;
                time += time_reduction;
                if (P[i].rbt == 0) {
                    P[i].ct = time;
                    P[i].tat = P[i].ct - P[i].at;
                    P[i].wt = P[i].tat - P[i].bt;
                    completed++;
                }
            }
        }
        if (all_done) {
            time++;
        }
    }
    printf("\n=== Round Robin ===\n");
}

int main(){
    int n;
    printf("Enter number of proccesses : ");
    scanf("%d", &n);
    struct process P[n];

    //input all process along with burst time
    printf("=== Enter the process details === \n");
    for(int i=0; i<n; i++){
        printf("process no: ");
        scanf("%d", &P[i].p);
        printf("burst time (duration) : ");
        scanf("%d", &P[i].bt);
        printf("arrival time : ");
        scanf("%d", &P[i].at);
        printf("priority : ");
        scanf("%d", &P[i].priority);
        printf("------------------------\n");
    }

    int go = 1;
    while(go){
        int choice;
        printf("\nChoose the Scheduling Algorithm\n1. First Come First
Serve\n2. Shortest Remaining Time First(SJF pre-emptive)\n3. Shortest Job
First(non pre-emptive)\n4. Priority Scheduling(pre-emptive)\n5. Priority
Scheduling(non pre-emptive)\n6. Round Robin\n7. Exit\nEnter choice : ");
        scanf("%d", &choice);
        switch(choice){
            case 1:
                FCFS_Algo(n, P);
                break;
            case 2:
                SJF_Algo(n ,P, 1);
                break;
            case 3:
                SJF_Algo(n ,P, 0);
                break;
            case 4:
                PS_Algo(n ,P, 1);
                break;
            case 5:
                PS_Algo(n ,P, 0);
                break;
            case 6:
                RR_Algo(n, P);
                break;
            case 7:
                go = 0;
                _exit(0);
            default:
                printf("\nInvalid choice !");
        }

        //display the process details
        printf("\n=============================================================
===============================\n");
        printf("| %-10s | %-10s | %-10s | %-10s | %-10s | %-10s | %-10s |",
"Process", "Priority", "BT", "AT", "WT", "TAT", "CT");
        printf("\n+------------+------------+------------+------------+--------
----+------------+------------+\n");
        for(int i=0; i<n; i++){
            printf("| P%-9d | %-10d | %-10d | %-10d | %-10d | %-10d | %-10d
|\n", P[i].p, P[i].priority,P[i].bt, P[i].at, P[i].wt, P[i].tat, P[i].ct);
        }
        printf("===============================================================
=============================\n");
        float avgW = 0, avgT = 0, avgC = 0;
        for(int i=0; i<n; i++){
            avgW += P[i].wt;
            avgT += P[i].tat;
            avgC += P[i].ct;
        }
        avgW = avgW / n;
        avgT = avgT / n;
        avgC = avgC / n;
        printf("\nAverage Waiting time      = %f", avgW);
        printf("\nAverage Turn Around Time = %f", avgT);
        printf("\nAverage Completion Time  = %f", avgC);
        printf("\n==========================================\n");
    }



    return 0;
}
```

```
Output:

Enter number of proccesses : 5
=== Enter the process details ===
process no: 1
burst time (duration) : 4
arrival time : 9
priority : 1
------------------------
process no: 2
burst time (duration) : 4
arrival time : 3
priority : 2
------------------------
process no: 3
burst time (duration) : 8
arrival time : 0
priority : 3
------------------------
process no: 4
burst time (duration) : 6
arrival time : 1
priority : 4
------------------------
process no: 5
burst time (duration) : 6
arrival time : 12
priority : 2
------------------------

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 1

=== First Come First Serve ===
===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 0          | 8          |
8          |
| P4         | 4          | 6          | 1          | 8          | 14         |
15         |
| P2         | 2          | 4          | 3          | 14         | 18         |
21         |
| P1         | 1          | 4          | 9          | 18         | 22         |
31         |
| P5         | 2          | 6          | 12         | 22         | 28         |
40         |
===============================================================================
=============

Average Waiting time      = 12.400000
Average Turn Around Time = 18.000000
Average Completion Time  = 23.000000
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 3

=== Shortest Job First ===

===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 0          | 8          |
8          |
| P4         | 4          | 6          | 1          | 15         | 21         |
22         |
| P2         | 2          | 4          | 3          | 5          | 9          |
12         |
| P1         | 1          | 4          | 9          | 3          | 7          |
16         |
| P5         | 2          | 6          | 12         | 10         | 16         |
28         |
===============================================================================
=============
Average Waiting time     = 6.600000
Average Turn Around Time = 12.200000
Average Completion Time  = 17.200001
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 2

=== Shortest Remaining Time First (SJF Preemptive) ===

===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 20         | 28         |
28         |
| P4         | 4          | 6          | 1          | 0          | 6          |
7          |
| P2         | 2          | 4          | 3          | 4          | 8          |
11         |
| P1         | 1          | 4          | 9          | 2          | 6          |
15         |
| P5         | 2          | 6          | 12         | 3          | 9          |
21         |
===============================================================================
=============

Average Waiting time     = 5.800000
Average Turn Around Time = 11.400000
Average Completion Time  = 16.400000
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 4

=== Priority Scheduling (Preemptive) ===
===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 14         | 22         |
22         |
| P4         | 4          | 6          | 1          | 21         | 27         |
28         |
| P2         | 2          | 4          | 3          | 0          | 4          |
7          |
| P1         | 1          | 4          | 9          | 0          | 4          |
13         |
| P5         | 2          | 6          | 12         | 1          | 7          |
19         |
===============================================================================
=============

Average Waiting time     = 7.200000
Average Turn Around Time = 12.800000
Average Completion Time  = 17.799999
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 5

=== Priority Scheduling (Non Preemptive) ===

===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 0          | 8          |
8          |
| P4         | 4          | 6          | 1          | 21         | 27         |
28         |
| P2         | 2          | 4          | 3          | 5          | 9          |
12         |
| P1         | 1          | 4          | 9          | 3          | 7          |
16         |
| P5         | 2          | 6          | 12         | 4          | 10         |
22         |
===============================================================================
=============
Average Waiting time     = 6.600000
Average Turn Around Time = 12.200000
Average Completion Time  = 17.200001
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 6
Time quantum : 3

=== Round Robin ===

===============================================================================
=============
| Process    | Priority   | BT         | AT         | WT         | TAT        |
CT         |
+------------+------------+------------+------------+------------+------------
+------------+
| P3         | 3          | 8          | 0          | 20         | 28         |
28         |
| P4         | 4          | 6          | 1          | 14         | 20         |
21         |
| P2         | 2          | 4          | 3          | 15         | 19         |
22         |
| P1         | 1          | 4          | 9          | 10         | 14         |
23         |
| P5         | 2          | 6          | 12         | 8          | 14         |
26         |
===============================================================================
=============

Average Waiting time     = 13.400000
Average Turn Around Time = 19.000000
Average Completion Time  = 24.000000
==========================================

Choose the Scheduling Algorithm
1. First Come First Serve
2. Shortest Remaining Time First(SJF pre-emptive)
3. Shortest Job First(non pre-emptive)
4. Priority Scheduling(pre-emptive)
5. Priority Scheduling(non pre-emptive)
6. Round Robin
7. Exit
Enter choice : 7
```