OS LAB ASSIGNMENT 5

Name :Mahesh Jagtap
Reg No. 24MCS1017

1) Write a C program to support the OS to do the short term scheduling using the following algorithms.

a) Round Robin (quantum 2 sec)
b) Priority Scheduling - Preemptive
c) Priority Scheduling – Non preemptive

Process	Arrival Time	CPU Burst
P1	0	5
P2	4	4
P3	3	7
P4	6	3
P5	7	1

Prepare a Gantt chart and calculate the Average Waiting Time and the Turnaround Time. Display which algorithm improves the efficiency for this group of processes.

Note: The output must have your register number and name

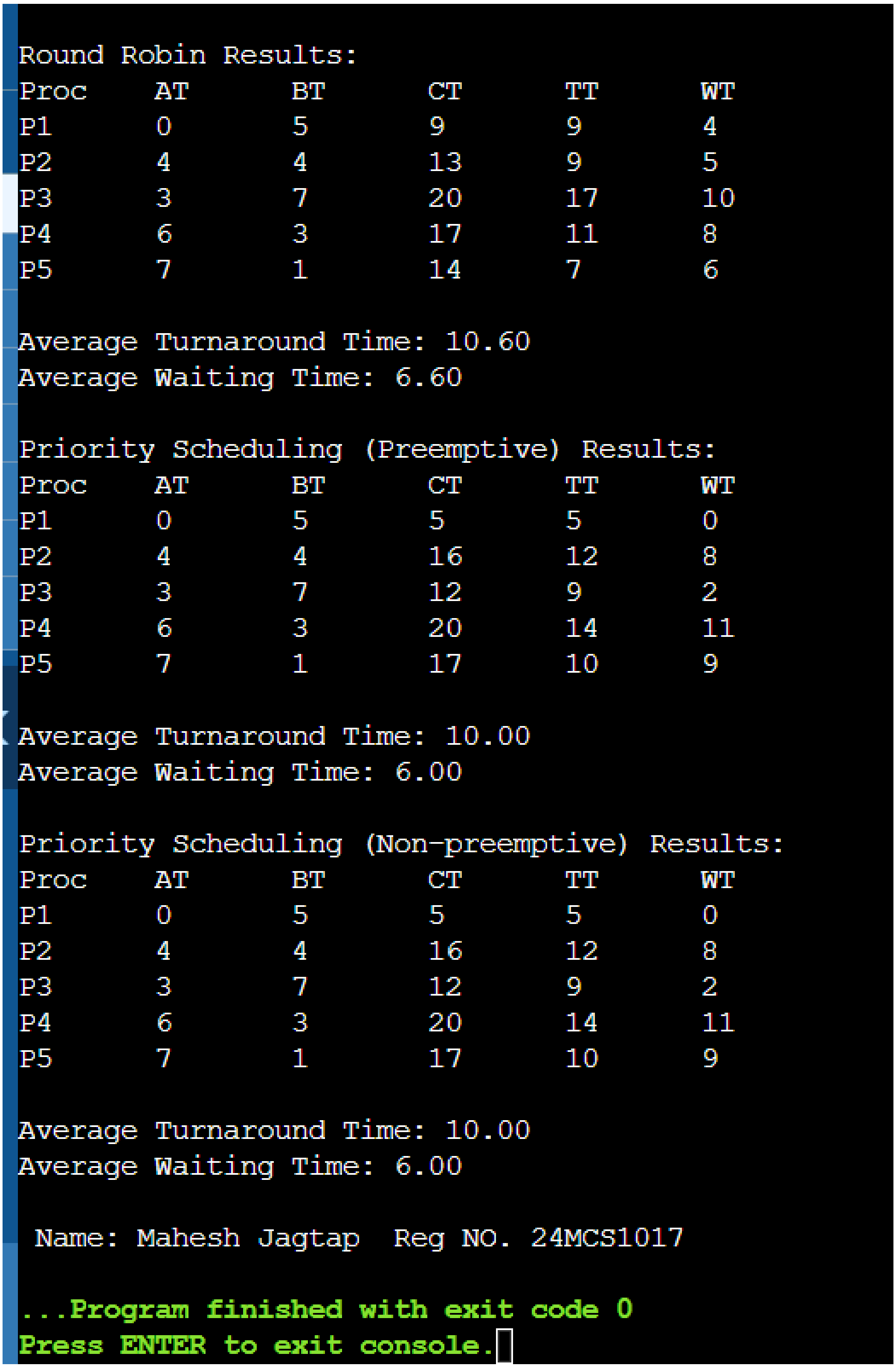
Code:

```
#include <stdio.h>

#include <limits.h>


#define MAX 10


typedef struct {

    int id, at, bt, rt, ct, wt, tt, pr; // Arrival Time, Burst Time, Remaining Time, Completion Time, Waiting Time, Turnaround Time, Priority
```

```
} Proc;

// Function to copy process data
void copyProcs(Proc src[], Proc dest[], int n) {
    for (int i = 0; i < n; i++) {
        dest[i] = src[i];
    }
}

// Function to calculate and display averages
void displayAverages(Proc procs[], int n, const char* algo) {
    int totalTAT = 0, totalWT = 0;
    printf("\n%s Results:\n", algo);
    printf("Proc\tAT\tBT\tCT\tTT\tWT\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n", procs[i].id, procs[i].at,
               procs[i].bt, procs[i].ct, procs[i].tt, procs[i].wt);
        totalTAT += procs[i].tt;
        totalWT += procs[i].wt;
    }
    printf("\nAverage Turnaround Time: %.2f", (float)totalTAT / n);
    printf("\nAverage Waiting Time: %.2f\n", (float)totalWT / n);
}

// Function for Round Robin Scheduling
void roundRobin(Proc procs[], int n, int tq) {
    int t = 0, c = 0, q[MAX], f = 0, r = 0, inQueue[MAX] = {0}, added[MAX] = {0};

    for (int i = 0; i < n; i++) {
        if (procs[i].at == 0) {
            q[r++] = i;
        inQueue[i] = 1;
        added[i] = 1;
    }
}

while (c < n) {
    if (f == r) {
        int nextAt = INT_MAX;
        for (int i = 0; i < n; i++) {
            if (!added[i] && procs[i].at < nextAt) {
                nextAt = procs[i].at;
            }
        }
        if (nextAt == INT_MAX) break;
        t = nextAt;
        for (int i = 0; i < n; i++) {
            if (procs[i].at <= t && !added[i]) {
                q[r++] = i;
                inQueue[i] = 1;
                added[i] = 1;
            }
        }
        continue;
    }

    int idx = q[f++];
    inQueue[idx] = 0;
    int exec = (procs[idx].rt < tq) ? procs[idx].rt : tq;
    t += exec;
    procs[idx].rt -= exec;
```

```
        for (int i = 0; i < n; i++) {
            if (!added[i] && procs[i].at <= t) {
                q[r++] = i;
                inQueue[i] = 1;
                added[i] = 1;
            }
        }

        if (procs[idx].rt == 0) {
            procs[idx].ct = t;
            procs[idx].tt = procs[idx].ct - procs[idx].at;
            procs[idx].wt = procs[idx].tt - procs[idx].bt;
            c++;
        } else {
            q[r++] = idx;
            inQueue[idx] = 1;
        }
    }
}

// Function for Preemptive Priority Scheduling
void priorityPreemptive(Proc procs[], int n) {
    int t = 0, c = 0, minPrIndex;
    int completed[MAX] = {0};

    while (c < n) {
        int minPr = INT_MAX;
        minPrIndex = -1;

        for (int i = 0; i < n; i++) {
            if (procs[i].at <= t && !completed[i] && procs[i].pr < minPr) {
                minPr = procs[i].pr;
                minPrIndex = i;
            }
        }

        if (minPrIndex == -1) {
            t++;
            continue;
        }

        procs[minPrIndex].rt--;
        t++;

        if (procs[minPrIndex].rt == 0) {
            procs[minPrIndex].ct = t;
            procs[minPrIndex].tt = procs[minPrIndex].ct - procs[minPrIndex].at;
            procs[minPrIndex].wt = procs[minPrIndex].tt - procs[minPrIndex].bt;
            completed[minPrIndex] = 1;
            c++;
        }
    }
}

// Function for Non-preemptive Priority Scheduling
void priorityNonPreemptive(Proc procs[], int n) {
    int t = 0, c = 0, minPrIndex;
    int completed[MAX] = {0};

    while (c < n) {
        int minPr = INT_MAX;
        minPrIndex = -1;
        for (int i = 0; i < n; i++) {
            if (procs[i].at <= t && !completed[i] && procs[i].pr < minPr) {
                minPr = procs[i].pr;
                minPrIndex = i;
            }
        }

        if (minPrIndex == -1) {
            t++;
            continue;
        }

        t += procs[minPrIndex].bt;
        procs[minPrIndex].ct = t;
        procs[minPrIndex].tt = procs[minPrIndex].ct - procs[minPrIndex].at;
        procs[minPrIndex].wt = procs[minPrIndex].tt - procs[minPrIndex].bt;
        completed[minPrIndex] = 1;
        c++;
    }
}

int main() {
    int n = 5;
    Proc procs[MAX] = {
        {1, 0, 5, 5, 0, 0, 0, 1},
        {2, 4, 4, 4, 0, 0, 0, 3},
        {3, 3, 7, 7, 0, 0, 0, 2},
        {4, 6, 3, 3, 0, 0, 0, 5},
        {5, 7, 1, 1, 0, 0, 0, 4}
    };
    int tq = 2;

    // Round Robin Scheduling
    Proc procsRR[MAX];
    copyProcs(procs, procsRR, n);
    roundRobin(procsRR, n, tq);
    displayAverages(procsRR, n, "Round Robin");

    // Priority Scheduling (Preemptive)
    Proc procsPSP[MAX];
    copyProcs(procs, procsPSP, n);
    priorityPreemptive(procsPSP, n);
    displayAverages(procsPSP, n, "Priority Scheduling (Preemptive)");

    // Priority Scheduling (Non-preemptive)
    Proc procsPSNP[MAX];
    copyProcs(procs, procsPSNP, n);
    priorityNonPreemptive(procsPSNP, n);
    displayAverages(procsPSNP, n, "Priority Scheduling (Non-preemptive)");

    printf("\n Name: Mahesh Jagtap  Reg NO. 24MCS1017");

    return 0;
}
```

OUTPUT:



```
Round Robin Results:
Proc    AT      BT      CT      TT      WT
P1      0       5       9       9       4
P2      4       4       13      9       5
P3      3       7       20      17      10
P4      6       3       17      11      8
P5      7       1       14      7       6

Average Turnaround Time: 10.60
Average Waiting Time: 6.60

Priority Scheduling (Preemptive) Results:
Proc    AT      BT      CT      TT      WT
P1      0       5       5       5       0
P2      4       4       16      12      8
P3      3       7       12      9       2
P4      6       3       20      14      11
P5      7       1       17      10      9

Average Turnaround Time: 10.00
Average Waiting Time: 6.00

Priority Scheduling (Non-preemptive) Results:
Proc    AT      BT      CT      TT      WT
P1      0       5       5       5       0
P2      4       4       16      12      8
P3      3       7       12      9       2
P4      6       3       20      14      11
P5      7       1       17      10      9

Average Turnaround Time: 10.00
Average Waiting Time: 6.00

 Name: Mahesh Jagtap  Reg NO. 24MCS1017

...Program finished with exit code 0
Press ENTER to exit console.
```