NAME: M.HASSAN HAYAT
ROLL NO: DT-22010
COURSE: OS

LAB NO 2:

1) Implement the First Come First Serve code and paste the output below.

```c
#include <stdio.h>

int main() {
    int bt[20], wt[20], tat[20], i, n;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i, bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```

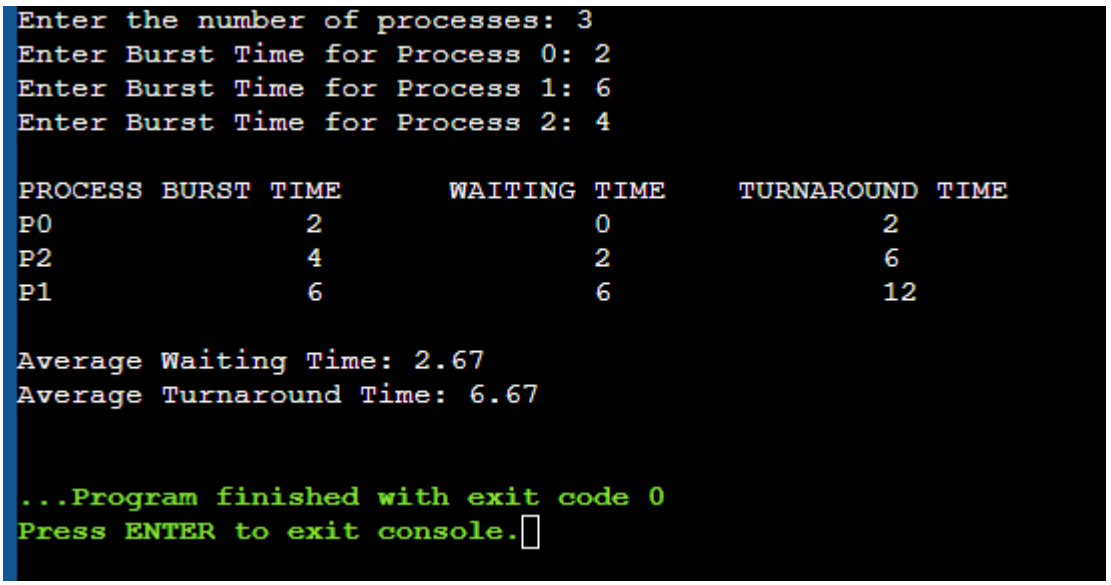
```
Enter the number of processes: 3
Enter Burst Time for Process 0: 2
Enter Burst Time for Process 1: 6
Enter Burst Time for Process 2: 4

PROCESS BURST TIME      WAITING TIME    TURNAROUND TIME
P0              2               0               2
P1              6               2               8
P2              4               8               12

Average Waiting Time: 3.33
Average Turnaround Time: 7.33
```

#include <stdio.h>

```
int main() {
    int bt[20], wt[20], tat[20], i, n;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND
TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i, bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```

2) Implement the Shortest Job First code and paste the output below.

```c
#include <stdio.h>

int main() {
    int bt[20], p[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

    for (i = 0; i < n - 1; i++) {
        for (k = i + 1; k < n; k++) {
            if (bt[i] > bt[k]) {
                temp = bt[i];
                bt[i] = bt[k];
                bt[k] = temp;
                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
            }
        }
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", p[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```

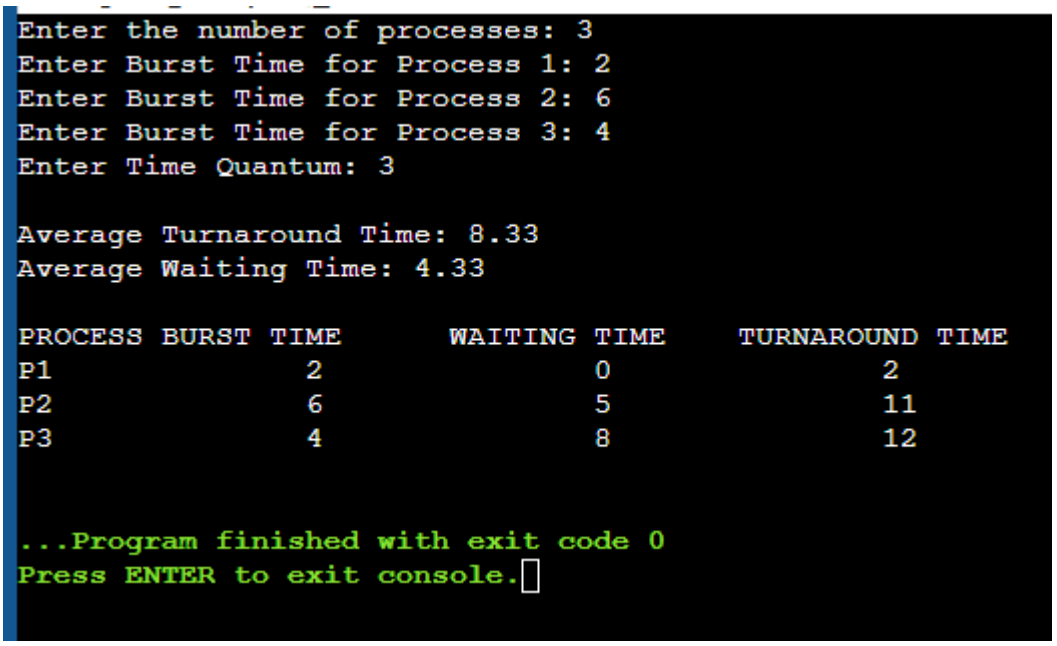



```
#include <stdio.h>

int main() {
    int bt[20], p[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time for Process %d: ", i);
        scanf("%d", &bt[i]);
    }

    for (i = 0; i < n - 1; i++) {
        for (k = i + 1; k < n; k++) {
            if (bt[i] > bt[k]) {
                temp = bt[i];
                bt[i] = bt[k];
                bt[k] = temp;
                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
```

```
            }
        }
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND
TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", p[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```

3) Implement the Round Robin code and paste the output below.

```c
#include <stdio.h>

int main() {
    int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
    float awt = 0, att = 0, temp = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i + 1);
        scanf("%d", &bu[i]);
        ct[i] = bu[i];
    }

    printf("Enter Time Quantum: ");
    scanf("%d", &t);

    max = bu[0];
    for (i = 1; i < n; i++)
        if (max < bu[i])
            max = bu[i];

    for (j = 0; j < (max / t) + 1; j++) {
        for (i = 0; i < n; i++) {
            if (bu[i] != 0) {
                if (bu[i] <= t) {
                    tat[i] = temp + bu[i];
                    temp += bu[i];
                    bu[i] = 0;
                } else {
                    bu[i] -= t;
                    temp += t;
                }
            }
        }
    }

    for (i = 0; i < n; i++) {
        wa[i] = tat[i] - ct[i];
        att += tat[i];
        awt += wa[i];
    }

    printf("\nAverage Turnaround Time: %.2f", att / n);
    printf("\nAverage Waiting Time: %.2f\n", awt / n);
    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, ct[i], wa[i], tat[i]);

    return 0;
}
```

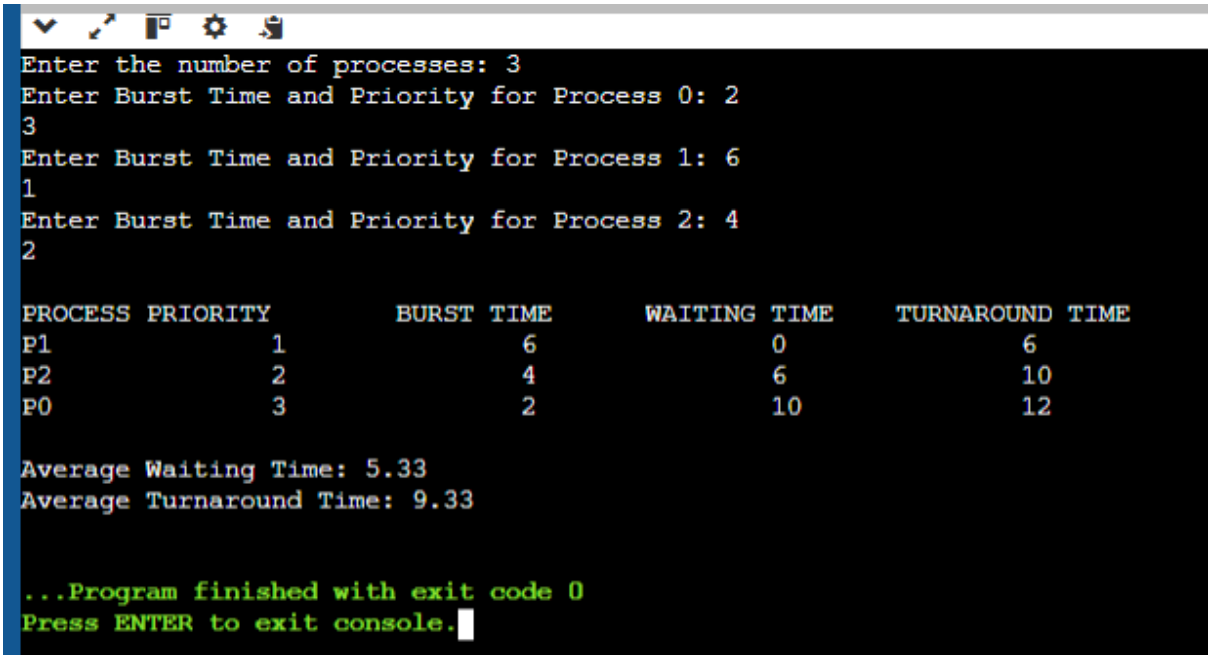



```
#include <stdio.h>

int main() {
    int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
    float awt = 0, att = 0, temp = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i + 1);
        scanf("%d", &bu[i]);
        ct[i] = bu[i];
    }

    printf("Enter Time Quantum: ");
    scanf("%d", &t);

    max = bu[0];
    for (i = 1; i < n; i++)
        if (max < bu[i])
            max = bu[i];
```

```
    for (j = 0; j < (max / t) + 1; j++) {
        for (i = 0; i < n; i++) {
            if (bu[i] != 0) {
                if (bu[i] <= t) {
                    tat[i] = temp + bu[i];
                    temp += bu[i];
                    bu[i] = 0;
                } else {
                    bu[i] -= t;
                    temp += t;
                }
            }
        }
    }

    for (i = 0; i < n; i++) {
        wa[i] = tat[i] - ct[i];
        att += tat[i];
        awt += wa[i];
    }

    printf("\nAverage Turnaround Time: %.2f", att / n);
    printf("\nAverage Waiting Time: %.2f\n", awt / n);
    printf("\nPROCESS\tBURST TIME\tWAITING TIME\tTURNAROUND
TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, ct[i], wa[i], tat[i]);

    return 0;
}
```

4) Implement the Priority Based Scheduling code and paste the output below.

```c
#include <stdio.h>

int main() {
    int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time and Priority for Process %d: ", i);
        scanf("%d%d", &bt[i], &pri[i]);
    }

    for (i = 0; i < n - 1; i++) {
        for (k = i + 1; k < n; k++) {
            if (pri[i] > pri[k]) {
                temp = pri[i];
                pri[i] = pri[k];
                pri[k] = temp;

                temp = bt[i];
                bt[i] = bt[k];
                bt[k] = temp;

                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
            }
        }
    }

    wtavg = wt[0] = 0;
    tatavg = tat[0] = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], pri[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```




```
#include <stdio.h>

int main() {
    int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter Burst Time and Priority for Process %d: ", i);
        scanf("%d%d", &bt[i], &pri[i]);
    }

    for (i = 0; i < n - 1; i++) {
        for (k = i + 1; k < n; k++) {
            if (pri[i] > pri[k]) {
                temp = pri[i];
                pri[i] = pri[k];
                pri[k] = temp;

                temp = bt[i];
                bt[i] = bt[k];
```

```
                bt[k] = temp;

                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
            }
        }
    }

    wtavg = wt[0] = 0;
    tatavg = tat[0] = bt[0];
    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
        wtavg += wt[i];
        tatavg += tat[i];
    }

    printf("\nPROCESS\tPRIORITY\tBURST TIME\tWAITING
TIME\tTURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], pri[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time: %.2f", wtavg / n);
    printf("\nAverage Turnaround Time: %.2f\n", tatavg / n);

    return 0;
}
```

Execute all scheduling algorithms on following data and find out the Average Waiting Time and
Average Turnaround Time of all scheduling algorithms and discuss your results.
(Quantum Value is 3)

1. First Come First Serve (FCFS)

Order of execution: **P0 -> P1 -> P2**

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P1	6	2	8
P2	4	8	12

AWT = (0 + 2 + 8) / 3 = **3.33**
TAT = (2 + 8 + 12) / 3 = **7.33**

2. Shortest Job First (SJF)

Order of execution**: P0 -> P2 -> P1**

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P2	4	2	6
P1	6	6	12

AWT = (0 + 2 + 6) / 3 **= 2.67**
TAT = **(**2 + 6 + 12) / 3 **= 6.67**

3. Round Robin (Quantum = 3)

Execution order: **P0 -> P1 -> P2 -> P1**

Process	Burst Time	Waiting Time	Turnaround Time
P0	2	0	2
P1	6	7	13
P2	4	3	7

AWT = (0 + 7 + 3) / 3 **= 3.33**
TAT = (2 + 13 + 7) / 3 **= 7.33**

4. Priority Scheduling

Order of execution (based on priority): **P1 -> P2 -> P0**

Process	Burst Time	Priority	Waiting Time	Turnaround Time
P1	6	1	0	6
P2	4	2	6	10
P0	2	3	10	12

AWT = (0 + 6 + 10) / 3 = **5.33**
TAT = (6 + 10 + 12) / 3 = **9.33**

Algorithm	Average Waiting Time	Average Turnaround Time
FCFS	3.33	7.33
SJF	2.67	6.67
Round Robin	3.33	7.33
Priority	5.33	9.33