ASSIGNMENT - III

TITLE :

Implement preemptive (Round Robin) and non-preemptive (FCFS, SJF, Priority) CPU scheduling algorithms, calculating and displaying waiting time, turnaround time, and CPU utilization for each.

NAME : Shinde Shubham Dnyandev, **DIV :** SY-B, **ROLL NO. :** 23107121.

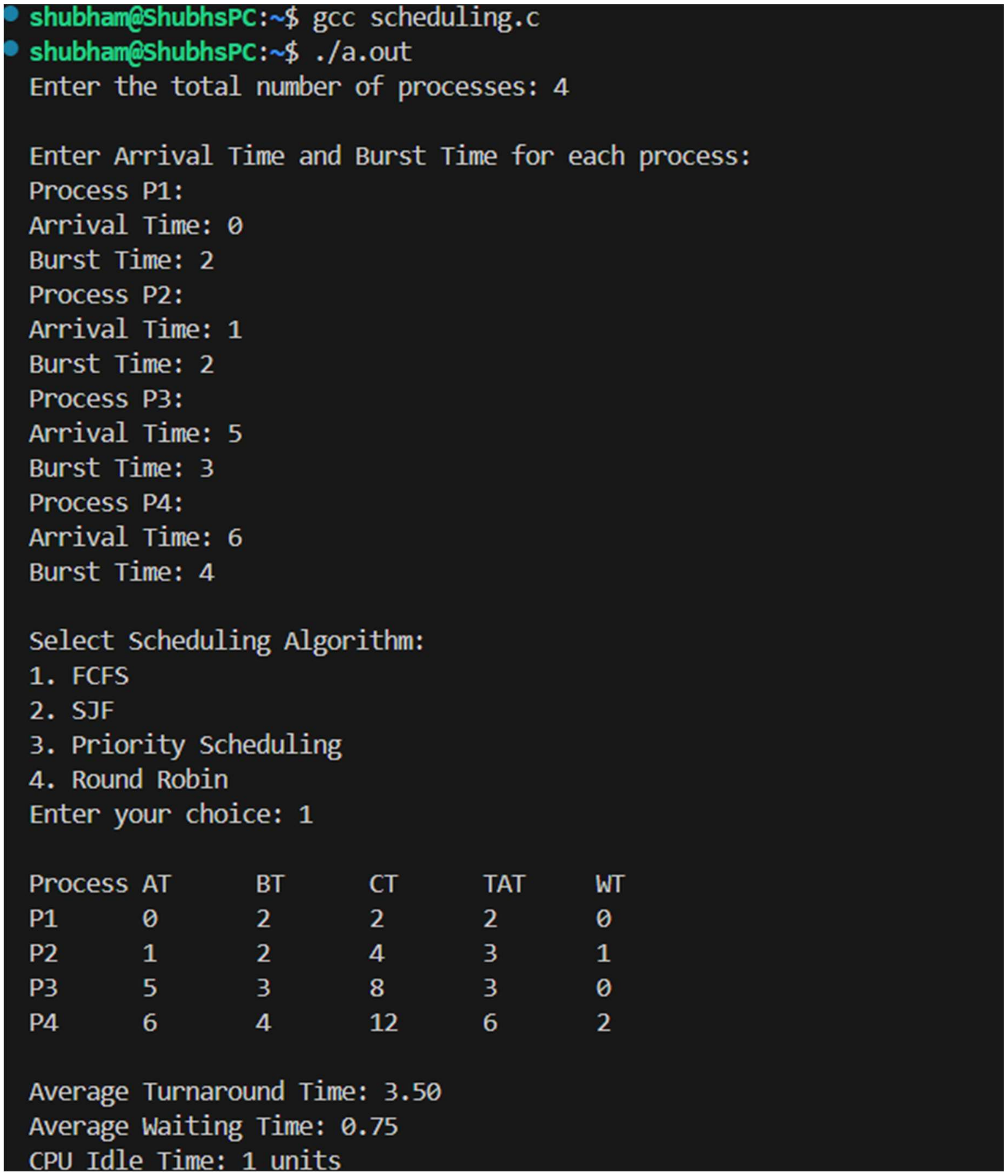
PROGRAM :

```
#include <stdio.h>

void sort(int n, int p[], int at[], int bt[], int pr[], int criterion) {
  int temp;

  for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {   //n-outer loop index - 1
        int swap = 0;

        if (criterion == 1 && (bt[j] > bt[j + 1] || (bt[j] == bt[j + 1] && at[j] > at[j + 1]))) {
            swap = 1;
        }

        else if (criterion == 2 && (pr[j] > pr[j + 1] ||
            (pr[j] == pr[j + 1] && at[j] > at[j + 1]))) {
            swap = 1;
        }

        if (swap) {
            temp = bt[j];
            bt[j] = bt[j + 1];
            bt[j + 1] = temp;

            temp = at[j];
            at[j] = at[j + 1];
            at[j + 1] = temp;

            if (pr != NULL) {
                temp = pr[j];
                pr[j] = pr[j + 1];
                pr[j + 1] = temp;
                }

                temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }
}

void completion_time(int n, int at[], int bt[], int ct[], int *idle_time) {
    int current_time = 0;
    *idle_time = 0;

    for (int i = 0; i < n; i++) {
        if (current_time < at[i]) {
            *idle_time += at[i] - current_time;
            current_time = at[i];
        }
        current_time += bt[i];
        ct[i] = current_time;
    }
}

void turnaround_time(int n, int at[], int ct[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = ct[i] - at[i];
    }
}

void waiting_time(int n, int tat[], int bt[], int wt[]) {
    for (int i = 0; i < n; i++) {
        wt[i] = tat[i] - bt[i];
    }
}

float avg_time(int n, int times[]) {
    float total = 0;
    for (int i = 0; i < n; i++) {
        total += times[i];
    }
    return total / n;
}
```

```
void fcfs(int n, int at[], int bt[], int wt[], int tat[], int ct[], int *idle_time) {
    completion_time(n, at, bt, ct, idle_time);
    turnaround_time(n, at, ct, tat);
    waiting_time(n, tat, bt, wt);
}

void sjf(int n, int p[], int at[], int bt[], int wt[], int tat[], int ct[], int *idle_time) {
    sort(n, p, at, bt, NULL, 1);
    completion_time(n, at, bt, ct, idle_time);
    turnaround_time(n, at, ct, tat);
    waiting_time(n, tat, bt, wt);
}

void priority(int n, int p[], int at[], int bt[], int pr[], int wt[], int tat[], int ct[], int *idle_time) {
    sort(n, p, at, bt, pr, 2);
    completion_time(n, at, bt, ct, idle_time);
    turnaround_time(n, at, ct, tat);
    waiting_time(n, tat, bt, wt);
}

void round_robin(int n, int at[], int bt[], int wt[], int tat[], int ct[], int time_quantum, int
*idle_time) {
    int remaining_bt[n], current_time = 0, completed = 0;
    *idle_time = 0;

    for (int i = 0; i < n; i++)
    remaining_bt[i] = bt[i];

    while (completed < n) {
        int idle = 1;
        for (int i = 0; i < n; i++) {
            if (remaining_bt[i] > 0 && at[i] <= current_time) {
                idle = 0;
                if (remaining_bt[i] > time_quantum) {
                    current_time += time_quantum;
                    remaining_bt[i] -= time_quantum;
                } else {
                    current_time += remaining_bt[i];
                    remaining_bt[i] = 0;
                    ct[i] = current_time;
                    completed++;
                }
            }
        }
        if (idle) current_time++;
    }

    turnaround_time(n, at, ct, tat);
    waiting_time(n, tat, bt, wt);
}

int main() {
    int n, choice, idle_time = 0, time_quantum;
    int at[10], bt[10], wt[10], tat[10], ct[10], p[10], pr[10];

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

    for (int i = 0; i < n; i++) {
        p[i] = i + 1;
    }

    printf("\nEnter Arrival Time and Burst Time for each process:\n");
    for (int i = 0; i < n; i++) {
        printf("Process P%d:\n", p[i]);
        printf("Arrival Time: ");
        scanf("%d", &at[i]);
        printf("Burst Time: ");
        scanf("%d", &bt[i]);
    }

    do{
    printf("\nSelect Scheduling Algorithm:\n");
    printf("1. FCFS\n2. SJF\n3. Priority Scheduling\n4. Round Robin\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            fcfs(n, at, bt, wt, tat, ct, &idle_time);
            break;

        case 2:
            sjf(n, p, at, bt, wt, tat, ct, &idle_time);
            break;
        case 3:
            printf("\nEnter Priority for each process:\n");
            for (int i = 0; i < n; i++) {
                printf("Priority of process P%d: ", p[i]);
                scanf("%d", &pr[i]);
            }
            priority(n, p, at, bt, pr, wt, tat, ct, &idle_time);
            break;

        case 4:
            printf("\nEnter Time Quantum: ");
            scanf("%d", &time_quantum);
            round_robin(n, at, bt, wt, tat, ct, time_quantum, &idle_time);
            break;

        default:
            printf("Invalid choice!\n");
            return 1;
    }
    }while(choice<1 || choice>4);

    printf("\nProcess\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], at[i], bt[i], ct[i], tat[i], wt[i]);
    }

    printf("\nAverage Turnaround Time: %.2f\n", avg_time(n, tat));
    printf("Average Waiting Time: %.2f\n", avg_time(n, wt));
    printf("CPU Idle Time: %d units\n", idle_time);

    return 0;
}
```

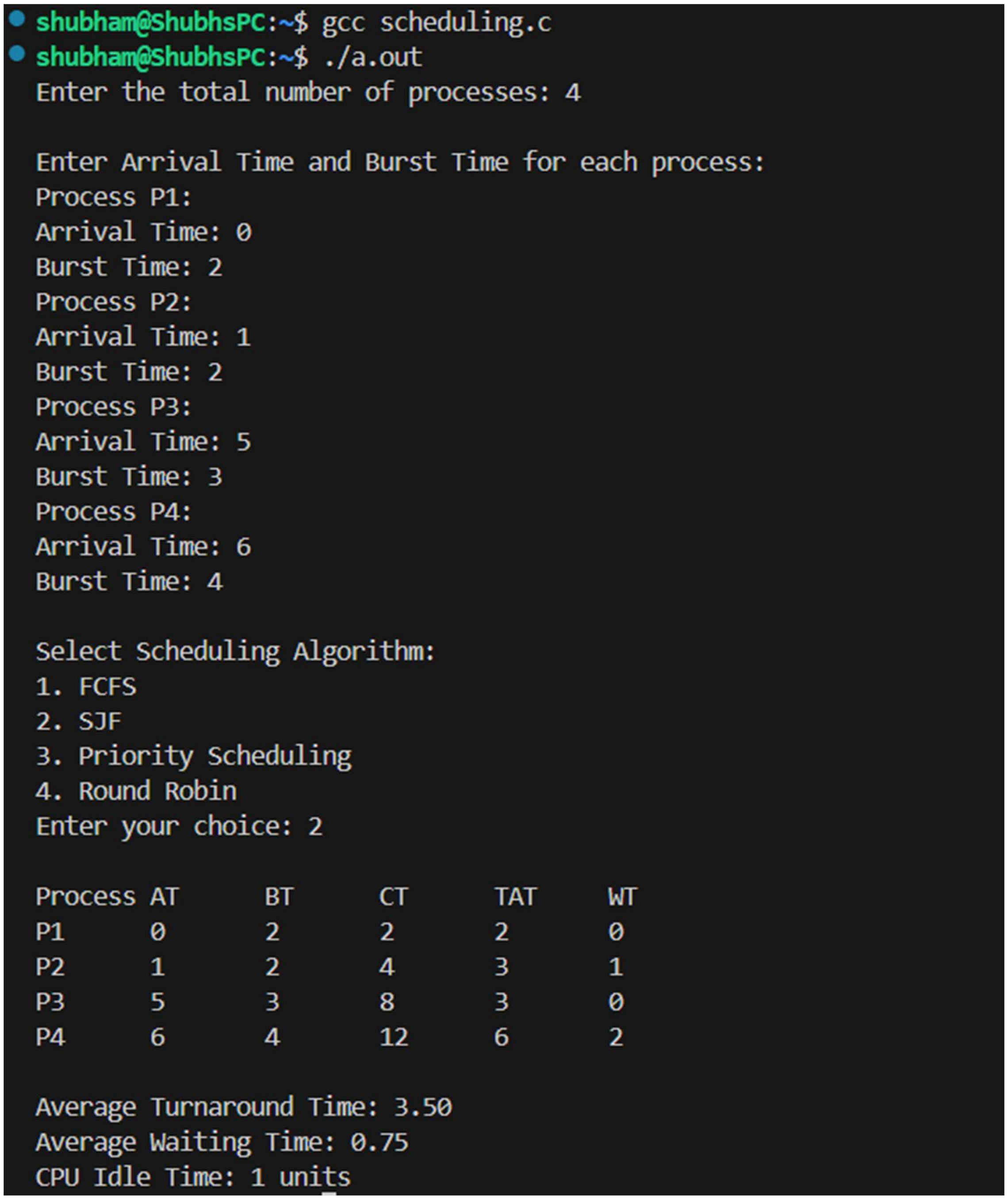
OUTPUT :

1)FCFS :



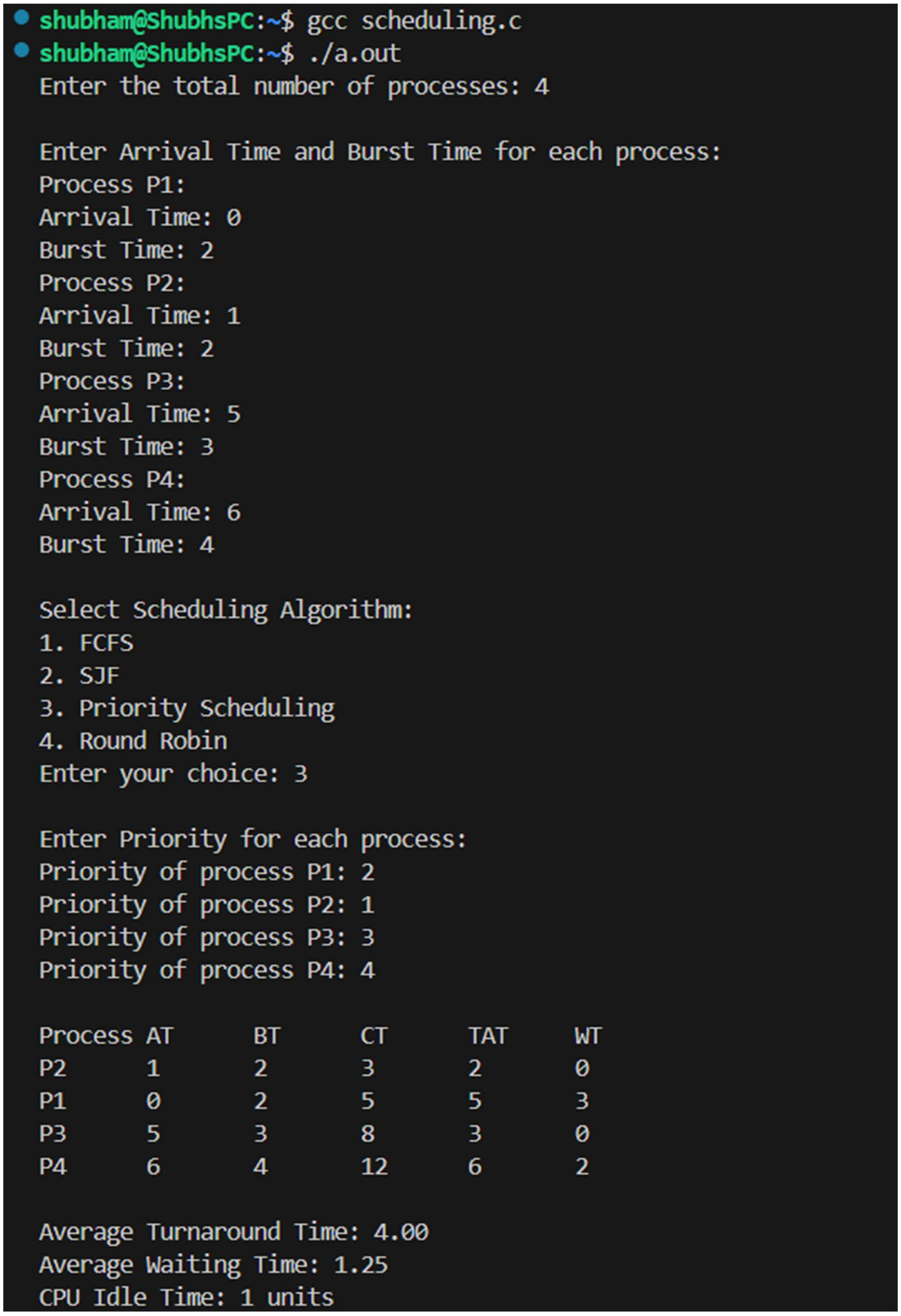
```
shubham@ShubhsPC:~$ gcc scheduling.c
shubham@ShubhsPC:~$ ./a.out
Enter the total number of processes: 4

Enter Arrival Time and Burst Time for each process:
Process P1:
Arrival Time: 0
Burst Time: 2
Process P2:
Arrival Time: 1
Burst Time: 2
Process P3:
Arrival Time: 5
Burst Time: 3
Process P4:
Arrival Time: 6
Burst Time: 4

Select Scheduling Algorithm:
1. FCFS
2. SJF
3. Priority Scheduling
4. Round Robin
Enter your choice: 1

Process AT      BT      CT      TAT     WT
P1      0       2       2       2       0
P2      1       2       4       3       1
P3      5       3       8       3       0
P4      6       4       12      6       2

Average Turnaround Time: 3.50
Average Waiting Time: 0.75
CPU Idle Time: 1 units
```

2)SJF :



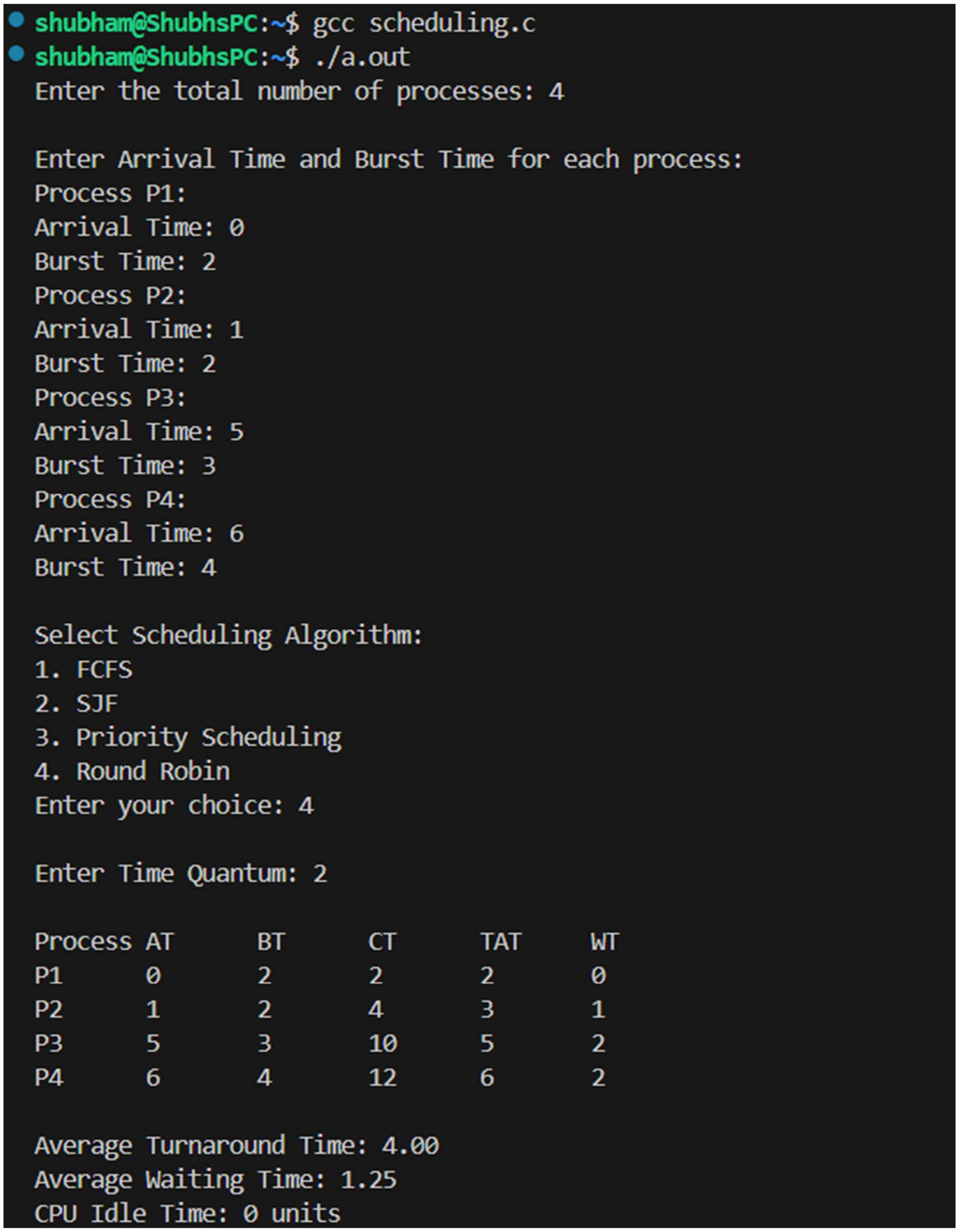
```
shubham@ShubhsPC:~$ gcc scheduling.c
shubham@ShubhsPC:~$ ./a.out
Enter the total number of processes: 4

Enter Arrival Time and Burst Time for each process:
Process P1:
Arrival Time: 0
Burst Time: 2
Process P2:
Arrival Time: 1
Burst Time: 2
Process P3:
Arrival Time: 5
Burst Time: 3
Process P4:
Arrival Time: 6
Burst Time: 4

Select Scheduling Algorithm:
1. FCFS
2. SJF
3. Priority Scheduling
4. Round Robin
Enter your choice: 2

Process AT      BT      CT      TAT     WT
P1      0       2       2       2       0
P2      1       2       4       3       1
P3      5       3       8       3       0
P4      6       4       12      6       2

Average Turnaround Time: 3.50
Average Waiting Time: 0.75
CPU Idle Time: 1 units
```

3)Priority :



```
shubham@ShubhsPC:~$ gcc scheduling.c
shubham@ShubhsPC:~$ ./a.out
Enter the total number of processes: 4

Enter Arrival Time and Burst Time for each process:
Process P1:
Arrival Time: 0
Burst Time: 2
Process P2:
Arrival Time: 1
Burst Time: 2
Process P3:
Arrival Time: 5
Burst Time: 3
Process P4:
Arrival Time: 6
Burst Time: 4

Select Scheduling Algorithm:
1. FCFS
2. SJF
3. Priority Scheduling
4. Round Robin
Enter your choice: 3

Enter Priority for each process:
Priority of process P1: 2
Priority of process P2: 1
Priority of process P3: 3
Priority of process P4: 4

Process AT      BT      CT      TAT     WT
P2      1       2       3       2       0
P1      0       2       5       5       3
P3      5       3       8       3       0
P4      6       4       12      6       2

Average Turnaround Time: 4.00
Average Waiting Time: 1.25
CPU Idle Time: 1 units
```

4)Round Robin :



```
shubham@ShubhsPC:~$ gcc scheduling.c
shubham@ShubhsPC:~$ ./a.out
Enter the total number of processes: 4

Enter Arrival Time and Burst Time for each process:
Process P1:
Arrival Time: 0
Burst Time: 2
Process P2:
Arrival Time: 1
Burst Time: 2
Process P3:
Arrival Time: 5
Burst Time: 3
Process P4:
Arrival Time: 6
Burst Time: 4

Select Scheduling Algorithm:
1. FCFS
2. SJF
3. Priority Scheduling
4. Round Robin
Enter your choice: 4

Enter Time Quantum: 2

Process AT      BT      CT      TAT     WT
P1      0       2       2       2       0
P2      1       2       4       3       1
P3      5       3       10      5       2
P4      6       4       12      6       2

Average Turnaround Time: 4.00
Average Waiting Time: 1.25
CPU Idle Time: 0 units
```