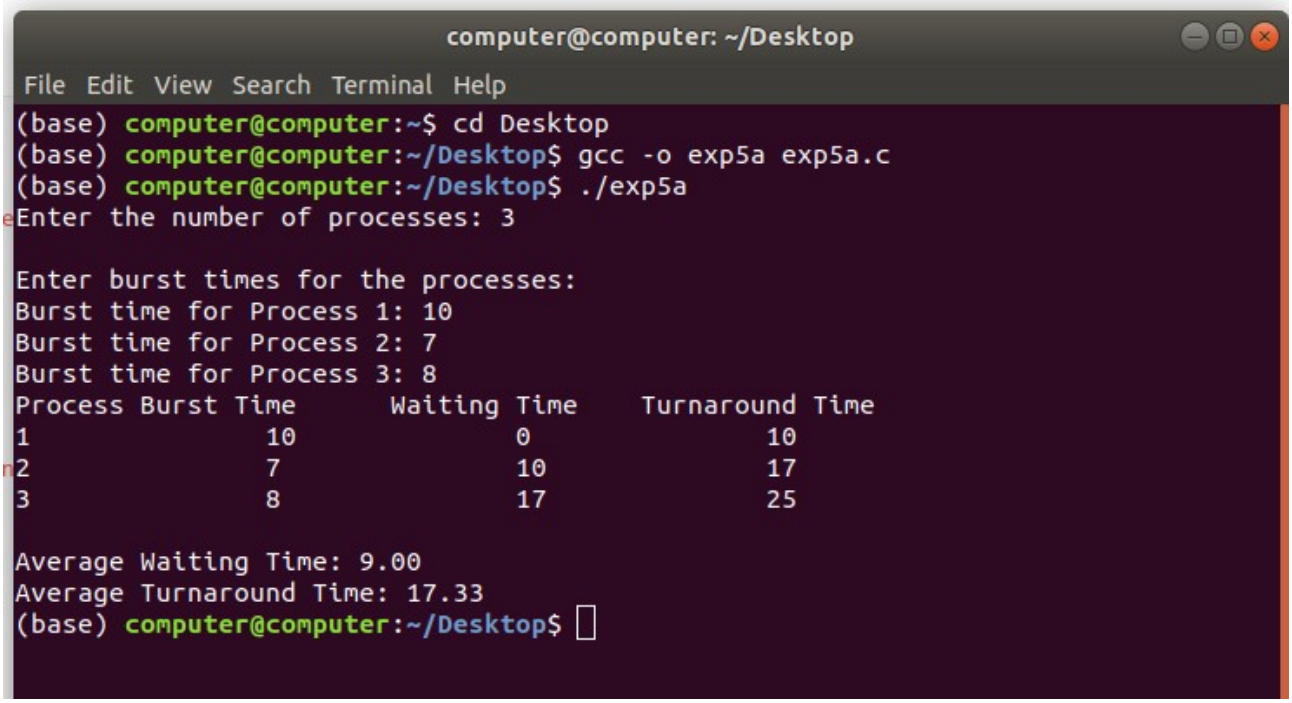
SOURCE CODE :

```
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0; // Waiting time for the first process is always 0

    for (int i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1]; /
    }
}
void findTurnaroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {

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

void findAverageTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt);
    findTurnaroundTime(processes, n, bt, wt, tat);

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time: %.2f\n", (float)total_wt / n);
    printf("Average Turnaround Time: %.2f\n", (float)total_tat / n);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int processes[n];     // Array for process IDs
    int burst_time[n];     // Array for burst times

    printf("\nEnter burst times for the processes:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;  // Process IDs starting from 1
        printf("Burst time for Process %d: ", processes[i]);
        scanf("%d", &burst_time[i]);
    }
 findAverageTime(processes, n, burst_time);
return 0;
}
```

OUTPUT :



SOURCE CODE :

```
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0; // Waiting time for the first process is always 0

    for (int i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1];
    }
}
void findTurnaroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}
void findAverageTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;
    findWaitingTime(processes, n, bt, wt);
    findTurnaroundTime(processes, n, bt, wt, tat);
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i], tat[i]);
    }
    printf("\nAverage Waiting Time: %.2f\n", (float)total_wt / n);
    printf("Average Turnaround Time: %.2f\n", (float)total_tat / n);
```

```
}

void sortProcessesByBurstTime(int processes[], int n, int bt[]) {
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (bt[i] > bt[j]) {

                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }
    }
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int processes[n];     // Array for process IDs
    int burst_time[n];     // Array for burst times

    printf("\nEnter burst times for the processes:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;  // Process IDs starting from 1
        printf("Burst time for Process %d: ", processes[i]);
        scanf("%d", &burst_time[i]);
    }
    sortProcessesByBurstTime(processes, n, burst_time);
    findAverageTime(processes, n, burst_time);
    return 0;
}
```

OUTPUT :