Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.

→ Priority

→Round Robin (Experiment with different quantum sizes for RR algorithm)

Input:

```c
#include <stdio.h>
#include <stdlib.h>


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


    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (priority[j] > priority[j + 1]) {

                int temp = priority[j];
                priority[j] = priority[j + 1];
                priority[j + 1] = temp;

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

    wt[0] = 0;


    for (int i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        total_wt += wt[i];
    }

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


    printf("\nProcess\t Burst Time\t Priority\t Waiting Time\t Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t %d\t\t %d\t\t %d\t\t %d\n", i + 1, bt[i], priority[i], wt[i], tat[i]);
    }
    printf("Average waiting time: %.2f\n", (float)total_wt / n);
    printf("Average turnaround time: %.2f\n", (float)total_tat / n);
}

void roundRobinScheduling(int n, int bt[], int quantum) {
    int remaining_bt[n], wt[n], tat[n], total_wt = 0, total_tat = 0;
```

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

    int t = 0;


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


    while (1) {
        int done = 1;


        for (int i = 0; i < n; i++) {
            if (remaining_bt[i] > 0) {
                done = 0;


                if (remaining_bt[i] > quantum) {
                    t += quantum;
                    remaining_bt[i] -= quantum;
                } else {
                    t += remaining_bt[i];
                    remaining_bt[i] = 0;


                    wt[i] = t - bt[i];


                    tat[i] = t;
                }
            }
        }
    if (done == 1)
            break;
    }


    printf("\nProcess\t Burst Time\t Waiting Time\t Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t %d\t\t %d\t\t %d\n", i + 1, bt[i], wt[i], tat[i]);
        total_wt += wt[i];
        total_tat += tat[i];
    }
```

```
    printf("Average turnaround time: %.2f\n", (float)total_tat / n);
}

int main() {
    int choice, n;

    do {
        printf("\nChoose the scheduling algorithm:\n");
        printf("1. Priority Scheduling (Non-preemptive)\n");
        printf("2. Round Robin Scheduling\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\nEnter the number of processes for Priority Scheduling: ");
                scanf("%d", &n);

                int *bt1 = (int *)malloc(n * sizeof(int));
                int *priority1 = (int *)malloc(n * sizeof(int));

                printf("Enter burst time and priority for each process:\n");
                for (int i = 0; i < n; i++) {
                    printf("Process %d: ", i + 1);
                    scanf("%d %d", &bt1[i], &priority1[i]);
                }

                priorityShedulingNonPreemptive(n, bt1, priority1);


                free(bt1);
                free(priority1);
                break;
            case 2:
                printf("\nEnter the number of processes for Round Robin Scheduling: ");
                scanf("%d", &n);

                int *bt2 = (int *)malloc(n * sizeof(int));
                int quantum;

                printf("Enter burst time for each process:\n");
                for (int i = 0; i < n; i++) {
                    printf("Process %d: ", i + 1);
                    scanf("%d", &bt2[i]);
                }

                printf("Enter time quantum for Round Robin Scheduling: ");
                scanf("%d", &quantum);
```

```
            for (int i = 0; i < n; i++) {
                printf("Process %d: ", i + 1);
                scanf("%d", &bt2[i]);
            }

            printf("Enter time quantum for Round Robin Scheduling: ");
            scanf("%d", &quantum);

            roundRobinScheduling(n, bt2, quantum);


            free(bt2);
            break;
        case 3:
            printf("\nExiting...\n");
            break;
        default:
            printf("\nInvalid choice. Please enter a valid option.\n");
    }
} while (choice != 3);

return 0;
```

OUTPUT:

```
Choose the scheduling algorithm:
1. Priority Scheduling
2. Round Robin Scheduling
3. Exit
Enter your choice: 1

Enter the number of processes for Priority Scheduling: 5
Enter burst time and priority for each process:
Process 1: 10
3
Process 2: 1
1
Process 3: 2
4
Process 4: 1
5
Process 5: 5
2

Process  Burst Time      Priority        Waiting Time    Turnaround Time
1               1               1               0       1
2               5               2               1       6
3               10              3               6       16
4               2               4               16      18
5               1               5               18      19
Average waiting time: 8.20
Average turnaround time: 12.00
```

```
Choose the scheduling algorithm:
1. Priority Scheduling
2. Round Robin Scheduling
3. Exit
Enter your choice: 2

Enter the number of processes for Round Robin Scheduling: 3
Enter burst time for each process:
Process 1: 1
Process 2: 6
Process 3: 10
Enter time quantum for Round Robin Scheduling: 2

Process  Burst Time      Waiting Time    Turnaround Time
1                1               0               1
2                6               5               11
3                10              7               17
Average waiting time: 4.00
Average turnaround time: 9.67
```