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

typedef struct {
char id[3];
int arrival_time;
int consultation_time;
int remaining_time;
int waiting_time;
int turnaround_time;
} Patient;

void reset_patients(Patient patients[], int n) {
for (int i = 0; i < n; i++) {
patients[i].remaining_time = patients[i].consultation_time;
patients[i].waiting_time = 0;
patients[i].turnaround_time = 0;
}
}
int select_paitent(int index,Patient patients[],int n,int current_time)
{
for (int i = index; i < n; i++) {
if (patients[i+1].arrival_time <= current_time && patients[i+1].remaining_time > 0) {
return i+1;
}
}
return index;
}
void round_robin(Patient patients[], int n, int time_quantum) {
int current_time = 0;
int completed = 0;
int i, index;

printf("Gantt Chart: ");

while (completed < n) {
index = 0;
index = select_paitent(index,patients,n,current_time);


if (index == -1) {
current_time++;
continue;
}

if (patients[index].remaining_time > time_quantum) {
current_time += time_quantum;
patients[index].remaining_time -= time_quantum;
printf("%s (%d-%d) -> ", patients[index].id, current_time - time_quantum, current_time);
} else {
```

```
current_time += patients[index].remaining_time;
// patients[index].remaining_time = 0;
patients[index].turnaround_time = current_time - patients[index].arrival_time;
patients[index].waiting_time = patients[index].turnaround_time - patients[index].consultation_time;
completed++;
printf("%s (%d-%d) -> ", patients[index].id, current_time - patients[index].remaining_time, current_time);
patients[index].remaining_time = 0;
}
}

printf("\n");

float total_waiting_time = 0;
float total_turnaround_time = 0;

for (i = 0; i < n; i++) {
total_waiting_time += patients[i].waiting_time;
total_turnaround_time += patients[i].turnaround_time;
}

printf("\nAverage Waiting Time: %.2f", total_waiting_time / n);
printf("\nAverage Turnaround Time: %.2f\n", total_turnaround_time / n);
}

void round_robin_with_emergency(Patient patients[], int n, int time_quantum, Patient emergency) {
int current_time = 0;
int completed = 0;
int i, index;
int emergency_handled = 0;

printf("Gantt Chart: ");

while (completed < n || !emergency_handled) {
if (!emergency_handled && current_time >= emergency.arrival_time && emergency.remaining_time > 0) {
if (emergency.remaining_time > time_quantum) {
current_time += time_quantum;
emergency.remaining_time -= time_quantum;
printf("PE (%d-%d) -> ", current_time - time_quantum, current_time);
} else {
current_time += emergency.remaining_time;
emergency.remaining_time = 0;
printf("PE (%d-%d) -> ", current_time - emergency.consultation_time, current_time);
emergency_handled = 1;
}
continue;
}

index = -1;

for (i = 0; i < n; i++) {
```

```
if (patients[i].arrival_time <= current_time && patients[i].remaining_time > 0) {
index = i;
break;
}
}

if (index == -1) {
current_time++;
continue;
}

if (patients[index].remaining_time > time_quantum) {
current_time += time_quantum;
patients[index].remaining_time -= time_quantum;
printf("%s (%d-%d) -> ", patients[index].id, current_time - time_quantum, current_time);
} else {
current_time += patients[index].remaining_time;
patients[index].remaining_time = 0;
patients[index].turnaround_time = current_time - patients[index].arrival_time;
patients[index].waiting_time = patients[index].turnaround_time - patients[index].consultation_time;
completed++;
printf("%s (%d-%d) -> ", patients[index].id, current_time - patients[index].consultation_time, current_time);
}
}

printf("\n");

float total_waiting_time = 0;
float total_turnaround_time = 0;

for (i = 0; i < n; i++) {
total_waiting_time += patients[i].waiting_time;
total_turnaround_time += patients[i].turnaround_time;
}

printf("\nAverage Waiting Time: %.2f", total_waiting_time / n);
printf("\nAverage Turnaround Time: %.2f\n", total_turnaround_time / n);
}

int main() {
int n, time_quantum;
printf("Enter the number of patients: ");
scanf("%d", &n);

Patient patients[n];

for (int i = 0; i < n; i++) {
printf("Enter details for patient %d (ID Arrival Time Consultation Time): ", i + 1);
scanf("%s %d %d", patients[i].id, &patients[i].arrival_time, &patients[i].consultation_time);
patients[i].remaining_time = patients[i].consultation_time;
```

```
patients[i].waiting_time = 0;

patients[i].turnaround_time = 0;

}


printf("Enter the time quantum: ");

scanf("%d", &time_quantum);


printf("\nRound Robin Scheduling:\n");

round_robin(patients, n, time_quantum);


printf("\nRound Robin Scheduling with Emergency Patient:\n");

Patient emergency;

printf("Enter details for the emergency patient (ID Arrival Time Consultation Time): ");

scanf("%s %d %d", emergency.id, &emergency.arrival_time, &emergency.consultation_time);

emergency.remaining_time = emergency.consultation_time;

reset_patients(patients, n);

round_robin_with_emergency(patients, n, time_quantum, emergency);


return 0;

}
```

output:

```
cse8@wtl-23:~/Downloads$ ./a.out
Enter the number of patients: 3
Enter details for patient 1 (ID Arrival Time Consultation Time): 1 0 5
Enter details for patient 2 (ID Arrival Time Consultation Time): 2 1 4
Enter details for patient 3 (ID Arrival Time Consultation Time): 3 2 2
Enter the time quantum: 4

Round Robin Scheduling:
Gantt Chart: 1 (0-4) -> 2 (4-8) -> 3 (8-10) -> 1 (10-11) ->

Average Waiting Time: 5.00
Average Turnaround Time: 8.67

Round Robin Scheduling with Emergency Patient:
Enter details for the emergency patient (ID Arrival Time Consultation Time): 4 3 3
Gantt Chart: 1 (0-3) -> PE (3-6) -> 1 (6-8) -> 2 (8-12) -> 3 (12-14) ->

Average Waiting Time: 5
Average Turnaround Time: 8.5
```