Problem 5: Write a program to implement the round robin scheduling algorithm having variables time quantum and find the average turnaround time, waiting time, completion time and response time for overall process. Also Print Gantt chart for it.

Solution:

Source Code:

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>


typedef struct

{

   char process_name[3];

   int arrival_time;

   int burst_time;

   int complete_time;

   int turn_around_time;

   int wait_time;
```

```
    int response_time;
    int time_left;
} process;

int have_task(process arr[], int n){
    // printf("HERE");
    int have = 0,i;
    for(i=0; i<n;i++){
        if(arr[i].time_left > 0){
            have=1;
            break;
        }
    }
    return have;
}

void print_process_table(process arr[],int n){
    int i;
    puts(" _____ _____ _____ _____ _____ _____ _____ _____");
```

```
    puts("| Process Name | Arrival Time  | Burst Time | Complete Time | T
urn Around Time | Wait Time | Response Time |");

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

        puts("|_____|_____|_____|_____
_____|_____|_____|_____|");

        printf("|     %3s     |     %3d     |     %3d     |     %3d     |     %4d
  |    %3d    |     %3d     |\n",

        arr[i].process_name,arr[i].arrival_time,arr[i].burst_time,arr[i].comp
lete_time,arr[i].turn_around_time,arr[i].wait_time,arr[i].response_time);

    }

    puts("|_____|_____|_____|_____
_____|_____|_____|_____|");



}


void get_average(process arr[], int n){

    double tat=0,wt=0,rt=0;

    int i;

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

        tat += (double)arr[i].turn_around_time;

        wt += (double)arr[i].wait_time;

        rt += (double)arr[i].response_time;
```

```
    }

    printf("Total time to Complete = %3d        Average Time to Complete
 = %.3f\n",arr[n-1].complete_time,(double)arr[n-
1].complete_time/(double)n);

    printf("Total Turn Around Time = %.3f      Average Turn Around Tim
e = %.3f\n",tat,tat/(double)n);

    printf("Total Waiting Time = %.3f          Average Waiting Time = %.3
f\n",wt,wt/(double)n);

    printf("Total Response Time = %.3f         Average Response Time = %
.3f\n",rt,rt/(double)n);

}



void gnatt(process arr[],int n,int time_quantum){

    int i,j,time=0,total_time=0;

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

        arr[i].time_left = arr[i].burst_time;

    }

    printf("\n|");

    i=0;

    while(have_task(arr,n)==1){

        if(arr[i].time_left>0){

            printf("%3s  ",arr[i].process_name);
```

```
            printf("|");
        }
        if(time_quantum<arr[i].time_left){
            time = time_quantum ? time_quantum<arr[i].time_left : arr[i].tim
e_left;
        }else{
            time = arr[i].time_left;
        }
        arr[i].time_left-=time;
        i++;
        i%=n;
    }
    printf("\n");
    for(i=0; i<n;i++){
        arr[i].time_left = arr[i].burst_time;
    }
    i=0;
    while(have_task(arr,n)==1){
        if(arr[i].time_left>0){
            printf("%2d",total_time);
```

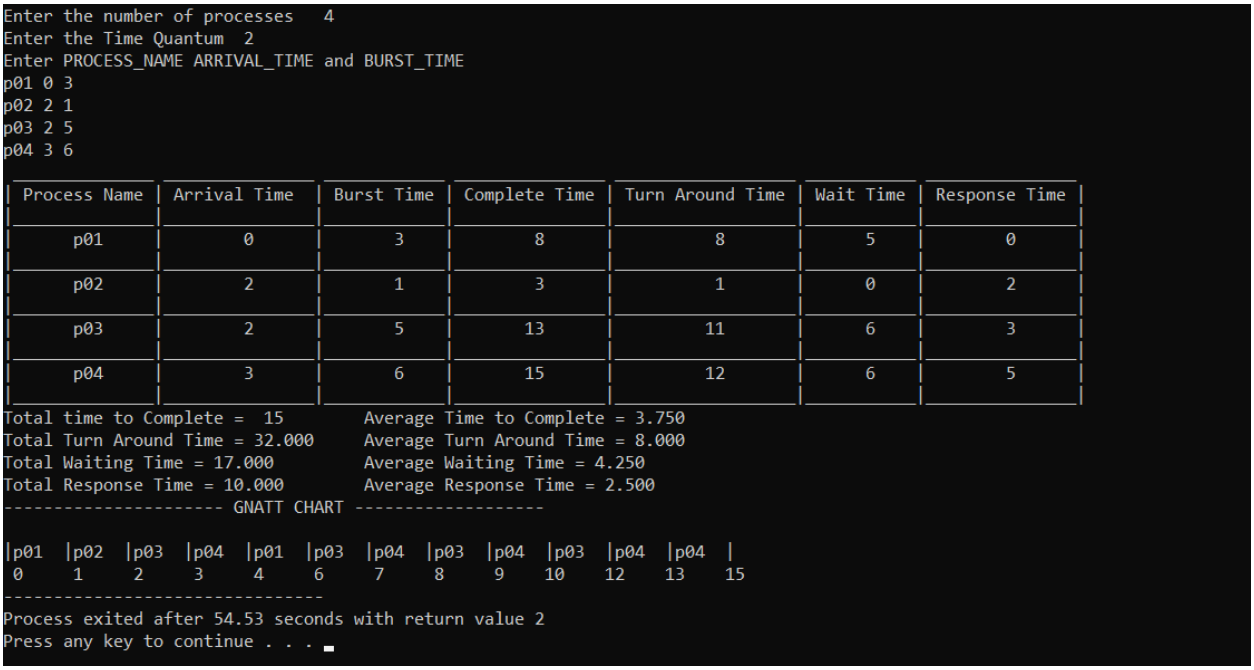
```
            printf("    ",arr[i].process_name);


        }
        if(time_quantum<arr[i].time_left){
            time = time_quantum ? time_quantum<arr[i].time_left : arr[i].tim
e_left;
        }else{
            time = arr[i].time_left;
        }
        arr[i].time_left-=time;
        total_time+=time;
        i++;
        i%=n;
    }
    printf("%d",total_time);


}


void main()
{
```

```
    int n =0,i, total_time=0, time_quantum;
    printf("Enter the number of processes\t");
    scanf("%d",&n);
    printf("Enter the Time Quantum\t");
    scanf("%d",&time_quantum);
    process arr[n];
    printf("Enter PROCESS_NAME ARRIVAL_TIME and BURST_TI
ME\n");
    for(i=0; i<n;i++)
    {
        scanf("%s %d %d",arr[i].process_name,&arr[i].arrival_time,&arr[i]
.burst_time);
        arr[i].time_left=arr[i].burst_time;
        // response time
        arr[i].response_time = total_time;
        if(arr[i].burst_time<time_quantum){
            total_time += arr[i].burst_time;
        }else{
            total_time += time_quantum;
        }
    }
```

```
    // complete time
    total_time=0;
    i=0;
    while(have_task(arr,n)==1){
        // printf("HERE IN WHILE");
        if(arr[i].time_left > 0){
            if(time_quantum<arr[i].time_left){
                total_time+=time_quantum;
                arr[i].time_left -= time_quantum;
            }else{
                total_time+=arr[i].time_left;
                arr[i].time_left -= arr[i].time_left;
            }
            if(arr[i].time_left==0){
                arr[i].complete_time = total_time;
                arr[i].turn_around_time = arr[i].complete_time-
arr[i].arrival_time;
                arr[i].wait_time = arr[i].turn_around_time-arr[i].burst_time;
            }
        }
```

```
        i++;
        i%=n;
    }
    print_process_table(arr,n);
    get_average(arr, n);
    printf("---------------------- GNATT CHART -------------------\n");
    gnatt(arr,n,time_quantum);
}
```

Output:



```
Enter the number of processes   4
Enter the Time Quantum  2
Enter PROCESS_NAME ARRIVAL_TIME and BURST_TIME
p01 0 3
p02 2 1
p03 2 5
p04 3 6
```

Process Name	Arrival Time	Burst Time	Complete Time	Turn Around Time	Wait Time	Response Time
p01	0	3	8	8	5	0
p02	2	1	3	1	0	2
p03	2	5	13	11	6	3
p04	3	6	15	12	6	5

```
Total time to Complete =  15        Average Time to Complete = 3.750
Total Turn Around Time = 32.000     Average Turn Around Time = 8.000
Total Waiting Time = 17.000         Average Waiting Time = 4.250
Total Response Time = 10.000        Average Response Time = 2.500
---------------------- GNATT CHART -------------------

|p01  |p02  |p03  |p04  |p01  |p03  |p04  |p03  |p04  |p03  |p04  |p04  |
 0     1     2     3     4     6     7     8     9     10    12    13    15
--------------------------------
Process exited after 54.53 seconds with return value 2
Press any key to continue . . .
```