

Problem 6: Write a program to implement the Non-preemptive priority scheduling algorithm 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>
#include<limits.h>

typedef struct
{
    char process_name[3];
    int arrival_time;
    int burst_time;
    int priority;
    int complete_time;
    int turn_around_time;
    int wait_time;
    int response_time;
    int done;
```

```
} process;
```

```
void print_process_table(process arr[],int n){
    int i;
    puts("_____");
    puts("| Process Name | Arrival Time | Burst Time | Complete Time | Turn 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].complete_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 Time = %.3f\n",
tat,tat/(double)n);

    printf("Total Waiting Time = %.3f    Average Waiting Time = %.3f\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 i,j;
    // upper row
    printf(" ");
    for(i=0; i<n;i++){
        for(j=0;j<arr[i].burst_time+1;j++) printf("__");
        printf(" ");
    }
    printf("\n");
    // middle row
    for(i=0;i<n;i++){
        for(j=0;j<arr[i].burst_time-1;j++){
            printf(" ");
        }
        printf("%3s",arr[i].process_name);
    }
}

```

```

        for(j=0;j<arr[i].burst_time;j++){
            printf(" ");
        }
        printf("|");
    }
    printf("\n");
    // lower row
    for(i=0; i<n;i++){
        for(j=0;j<arr[i].burst_time+1;j++) printf("__");
        printf("|");
    }
    printf("\n");
    printf("0");
    for(i=0; i<n; i++) {
        for(j=0; j<arr[i].burst_time+1; j++) printf(" ");
        if(arr[i].turn_around_time > 9) printf("\b");
        printf("%d", arr[i].turn_around_time);

    }
    printf("\n");
}

int completed(process arr[], int n){
    int i=0,flag=1;
    for(i=0;i<n;i++){

```

```
        if(arr[i].done==0){
            flag=0;
            break;
        }
    }
    return flag;
}
```

```
int best_process(process arr[], int n, int time){
    int ind=-1,i=0,priority=INT_MAX;
    for(i=0;i<n;i++){
        if(arr[i].arrival_time > time){
            break;
        }else{
            if(arr[i].done==0 && arr[i].priority<priority){
                priority=arr[i].priority;
                ind=i;
            }
        }
    }
    return ind;
}
```

```
void main()
{
```

```

int n =0,i, total_time=0,temp=0;
printf("Enter the number of processes\t");
scanf("%d",&n);
process arr[n], gnt[n];
printf("Enter PROCESS_NAME ARRIVAL_TIME BURST_TIME and PRIOR
ITY\n");
for(i=0; i<n;i++)
{
    scanf("%s %d %d %d",arr[i].process_name,&arr[i].arrival_time,&arr[i].burst
_time,&arr[i].priority);
    arr[i].done=0;
}

i=0;
while (completed(arr,n)!=1)
{
    temp=best_process(arr,n, total_time); //return index of that process to execute.
    if(temp!=-1){
        total_time++;
    }else{
        arr[temp].complete_time = total_time+arr[temp].burst_time;
        arr[temp].turn_around_time = arr[temp].complete_time-
arr[temp].arrival_time;
        arr[temp].response_time = total_time-arr[temp].arrival_time;
        arr[temp].wait_time = arr[temp].turn_around_time-arr[temp].burst_time;
        total_time += arr[temp].burst_time;
    }
}

```

```

        arr[temp].done=1;

        gnt[i++]=arr[temp];

    }

}

print_process_table(arr,n);

get_average(arr, n);

puts("----- GNATT CHART -----");

gnatt(gnt,n);

}

```

Output:

```

D:\os lab\Tanmay-Vig_19BCS061_p6.exe
Enter the number of processes 5
Enter PROCESS_NAME ARRIVAL_TIME BURST_TIME and PRIORITY
p01 0 3 3
p02 0 5 2
p03 3 4 1
p04 5 4 2
p05 6 7 5

```

Process Name	Arrival Time	Burst Time	Complete Time	Turn Around Time	Wait Time	Response Time
p01	0	3	16	16	13	13
p02	0	5	5	5	0	0
p03	3	4	9	6	2	2
p04	5	4	13	8	4	4
p05	6	7	23	17	10	10

```

Total time to Complete = 23      Average Time to Complete = 4.600
Total Turn Around Time = 52.000  Average Turn Around Time = 10.400
Total Waiting Time = 29.000      Average Waiting Time = 5.800
Total Response Time = 29.000     Average Response Time = 5.800
----- GNATT CHART -----

```

p02	p03	p04	p01	p05
0	5	6	8	16
				17

```

-----
Process exited after 144.4 seconds with return value 10
Press any key to continue . . .

```