Problem 8: Write a program to implement the Highest Response Ratio Next (Non-preemptive) algorithm and find the average turnaround time, waiting time, completion time and response time for overall process.

Answer:

Source Code

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<limits.h>

#include<float.h>


typedef struct

{

   char process_name[3];

   int arrival_time;

   int burst_time;

   float hrr;

   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_t
ime,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;
}

void update_hrr(process arr[], int n, int time){
    int i=0;
    for(i=0;i<n;i++){
        if(arr[i].done==0){
            arr[i].hrr=(float)(time-arr[i].arrival_time+arr[i].burst_time)/(float)arr[i].burst_time;
        }
    }
}
```

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

i=0;
while (completed(arr,n)!=1)
{
    update_hrr(arr,n,total_time);
    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: