OPERATING SYSTEMS

Lab-7: Priority and Round Robin

Name – Vinay Santosh Menon
Registration Number - 20BAI1103

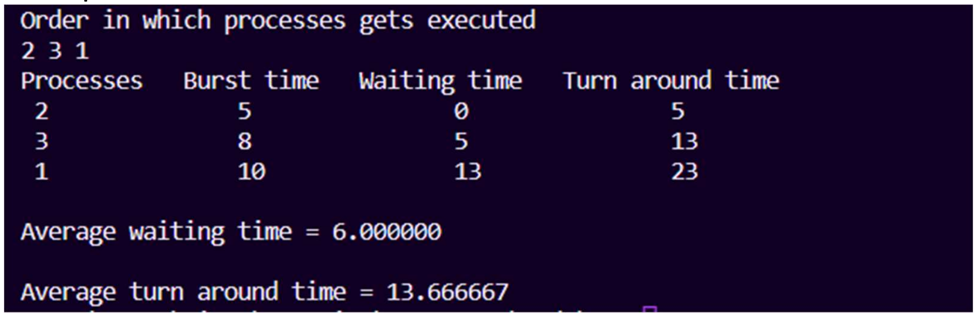
TASK:
To write a C code for demonstrating Priority and Round Robin with and without arrival time

Priority Without Arrival Time:

Code:

```
#include<stdio.h>

struct Process
{
    int pid;
    int bt;
    int priority;
};
int comparison(struct Process a, struct Process b)
{
    return (a.priority > b.priority);
}
void sort(struct Process procs[], int n) {
    for(int i=0; i<n; i++)
        for(int j=i+1; j<n; j++) {
            if (comparison(procs[i], procs[j])) {
                struct Process temp = procs[i];
                procs[i] = procs[j];
                procs[j] = temp;
            }
        }
}
void findWaitingTime(struct Process proc[], int n,
                     int wait_time[])
{
    wait_time[0] = 0;
    for (int  i = 1; i < n ; i++ )
        wait_time[i] =  proc[i-1].bt + wait_time[i-1] ;
}
void findTurnAroundTime(struct Process proc[], int n,
                         int wait_time[], int tat[])
{
    for (int  i = 0; i < n ; i++)
        tat[i] = proc[i].bt + wait_time[i];
}
void findavgTime(struct Process proc[], int n)
{
    int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
    findWaitingTime(proc, n, wait_time);
    findTurnAroundTime(proc, n, wait_time, tat);
    printf("\nProcesses   Burst time   Waiting time   Turn around time\n");
    for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wait_time[i];
        total_tat = total_tat + tat[i];
        printf(" %d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].bt,
wait_time[i], tat[i]);
    }
    printf("\nAverage waiting time = %f\n", (float)total_wt / (float)n);
    printf("\nAverage turn around time = %f\n", (float)total_tat / (float)n);
}

void priorityScheduling(struct Process proc[], int n)
{
    sort(proc, n);
    printf("Order in which processes gets executed \n");
    for (int  i = 0 ; i <  n; i++)
        printf("%d ", proc[i].pid);
    findavgTime(proc, n);
}
int main()
{
    struct Process proc[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}};
    int n = sizeof proc / sizeof proc[0];
    priorityScheduling(proc, n);
    return 0;
}
```

Example:

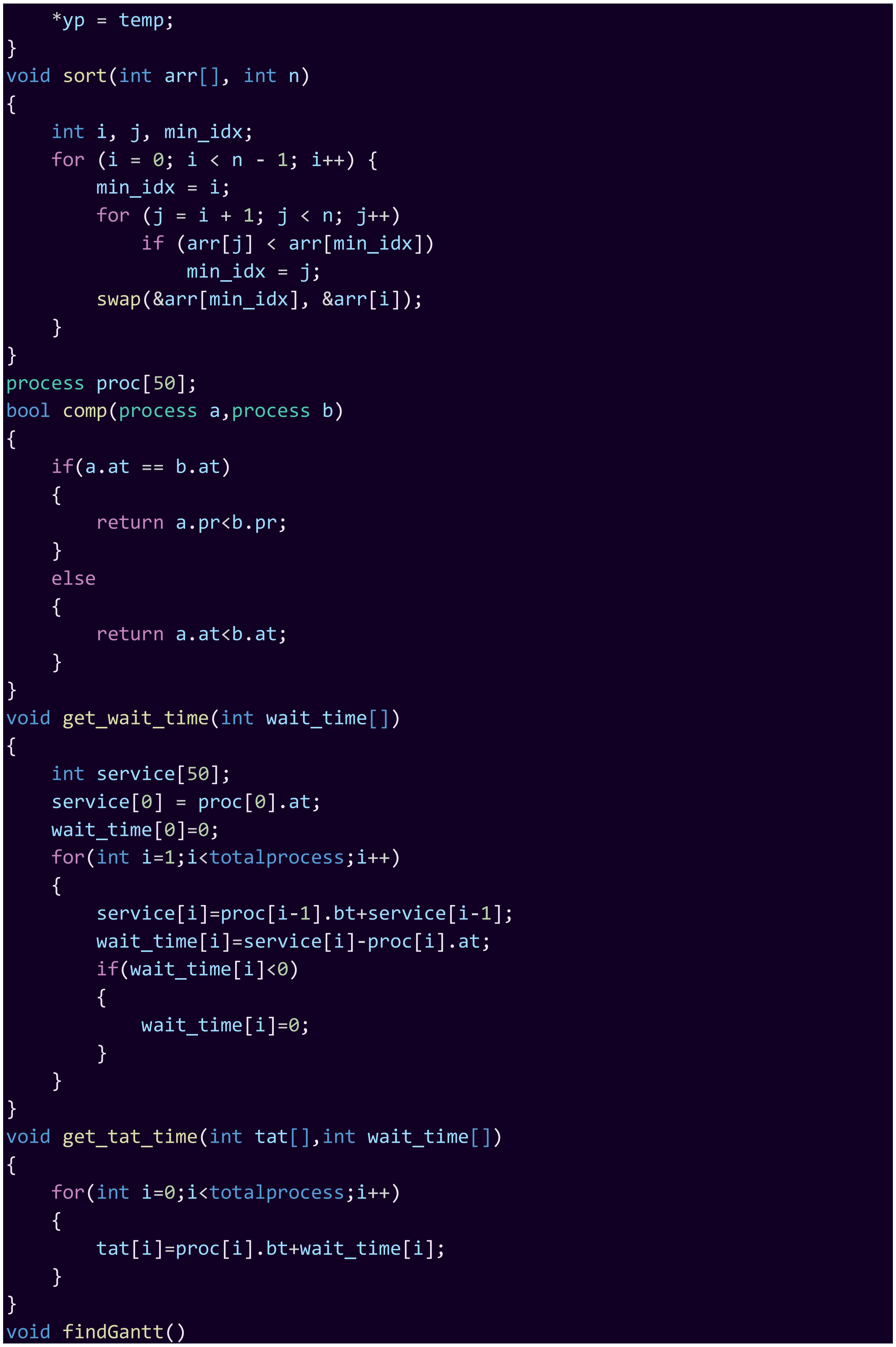


```
Order in which processes gets executed
2 3 1
Processes    Burst time    Waiting time    Turn around time
 2              5              0               5
 3              8              5               13
 1              10             13              23

Average waiting time = 6.000000

Average turn around time = 13.666667
```

Priority With Arrival Time:

Code:

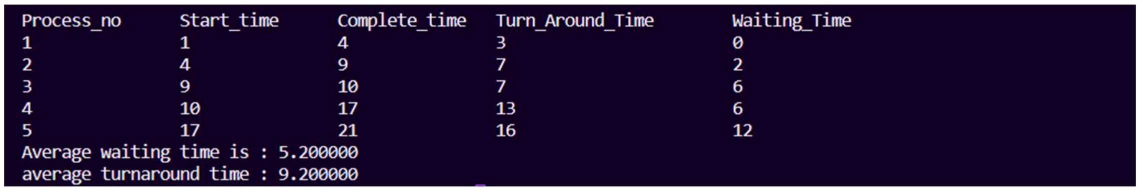
```
#include<stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#define totalprocess 5
typedef struct process
{
    int at,bt,pr,pno;
}process;
void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
```



```
    *yp = temp;
}
void sort(int arr[], int n)
{
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;
        swap(&arr[min_idx], &arr[i]);
    }
}
process proc[50];
bool comp(process a,process b)
{
    if(a.at == b.at)
    {
        return a.pr<b.pr;
    }
    else
    {
        return a.at<b.at;
    }
}
void get_wait_time(int wait_time[])
{
    int service[50];
    service[0] = proc[0].at;
    wait_time[0]=0;
    for(int i=1;i<totalprocess;i++)
    {
        service[i]=proc[i-1].bt+service[i-1];
        wait_time[i]=service[i]-proc[i].at;
        if(wait_time[i]<0)
        {
            wait_time[i]=0;
        }
    }
}
void get_tat_time(int tat[],int wait_time[])
{
    for(int i=0;i<totalprocess;i++)
    {
        tat[i]=proc[i].bt+wait_time[i];
    }
}
void findGantt()
```

```
{
    int wait_time[50],tat[50];
    int wavg=0,tavg=0;
    get_wait_time(wait_time);
    get_tat_time(tat,wait_time);
    int start_time[50],comp_time[50];
    start_time[0] = proc[0].at;
    comp_time[0]=start_time[0]+tat[0];
    for(int i=1;i<totalprocess;i++)
    {
        start_time[i]=comp_time[i-1];
        comp_time[i]=start_time[i]+tat[i]-wait_time[i];
    }
    printf("Process_no\tStart_time\tComplete_time\tTurn_Around_Time\tWaiting_Time\n");
    for(int i=0;i<totalprocess;i++)
    {
        wavg += wait_time[i];
        tavg += tat[i];
        printf("%d\t\t%d\t\t%d\t\t%d\t\t\t%d\n",proc[i].pno,start_time[i],comp_time[i],tat[i],wait_time[i]);
    }
    double w = wavg/(float)totalprocess;
    double t = tavg/(float)totalprocess;
    printf("Average waiting time is : ");
    printf("%f\n",w);
    printf("average turnaround time : ");
    printf("%f\n",t);
}
int main()
{
    int arrivaltime[] = { 1, 2, 3, 4, 5 };
    int bursttime[] = { 3, 5, 1, 7, 4 };
    int priority[] = { 3, 4, 1, 7, 8 };
    for(int i=0;i<totalprocess;i++)
    {
        proc[i].at=arrivaltime[i];
        proc[i].bt=bursttime[i];
        proc[i].pr=priority[i];
        proc[i].pno=i+1;
    }
    qsort(proc,totalprocess,sizeof(process),comp);
    findGantt();
    return 0;
}
```

Example:



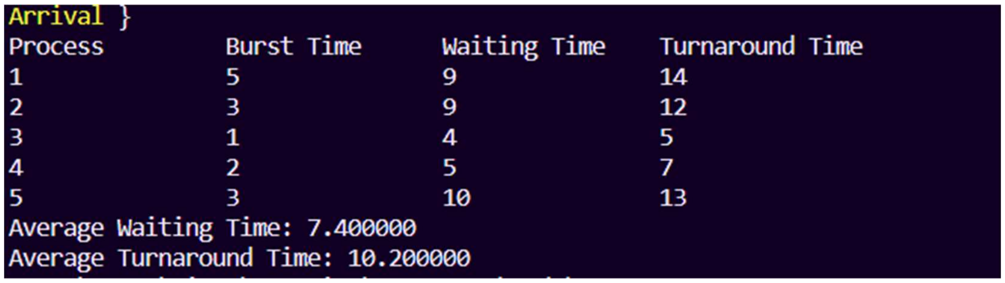
```
Process_no      Start_time      Complete_time   Turn_Around_Time       Waiting_Time
1               1               4               3                      0
2               4               9               7                      2
3               9               10              7                      6
4               10              17              13                     6
5               17              21              16                     12
Average waiting time is : 5.200000
average turnaround time : 9.200000
```

Round Robin Without Arrival Time:

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    int i,pid,status;
    int n = 5;
    int time,time_quantum = 2;
    int burst_time[10] = {5,3,1,2,3};
    int wait_time[10] ,turnaround_time[10],flag[10];
    int remaining_time[10] = {5,3,1,2,3};
    int total_wait_time = 0 ,total_turnaround_time = 0;
    float average_wait_time ,average_turnaround_time;
    for(i = 0; i < n; i++)
    {
        flag[i] = 1;
    }
    time = 0;
    while(1)
    {
        int done = 1;
        for(i = 0; i < n; i++){
            if(flag[i] == 1)
            {
                done = 0;
                if(remaining_time[i] > time_quantum)
                {
                    time += time_quantum;
                    remaining_time[i] -= time_quantum;
                }
                else
                {
                    time += remaining_time[i];
                    wait_time[i] = time - burst_time[i];
                    remaining_time[i] = 0;
                    flag[i] = 0;
                }
            }
        }
        if(done == 1)
        {
            break;
        }
    }
    for(i = 0; i < n; i++)
    {
        turnaround_time[i] = wait_time[i] + burst_time[i];
        total_wait_time += wait_time[i];
        total_turnaround_time += turnaround_time[i];
    }
    average_wait_time = (float)total_wait_time / (float)n;
    average_turnaround_time = (float)total_turnaround_time / (float)n;
    printf("Process\t\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for(i = 0; i < n; i++)
    {
        printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, burst_time[i], wait_time[i],
turnaround_time[i]);
    }
    printf("Average Waiting Time: %f\n", average_wait_time);
    printf("Average Turnaround Time: %f", average_turnaround_time);
    return 0;
}
```

Example:



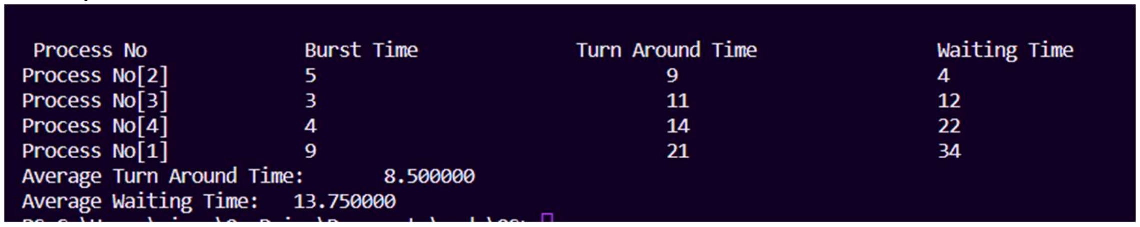
```
Arrival }
Process         Burst Time      Waiting Time    Turnaround Time
1               5               9               14
2               3               9               12
3               1               4               5
4               2               5               7
5               3               10              13
Average Waiting Time: 7.400000
Average Turnaround Time: 10.200000
```

Round Robin with Arrival Time:

Code:

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

void main()
{
    int i, n = 4, sum=0,count=0, y, quant = 5, wt=0, tat=0;
    int arrival_time[10] = {0,1,2,3};
    int burst_time[10] = {9,5,3,4};
    float avg_wt, avg_tat;
    int temp[10] = {9,5,3,4};
    y = n;
    printf("\n Process No \t\t Burst Time \t\t Turn Around Time \t\t Waiting 
Time ");
    for(sum=0, i = 0; y!=0; )
    {
    if(temp[i] <= quant && temp[i] > 0)
    {
        sum = sum + temp[i];
        temp[i] = 0;
        count=1;
        }
        else if(temp[i] > 0)
        {
            temp[i] = temp[i] - quant;
            sum = sum + quant;
        }
        if(temp[i]==0 && count==1)
        {
            y--;
            if(sum-arrival_time[i]-burst_time[i]<0)
            {
                wt += 0;
            }
            else
            {
                wt = wt+sum-arrival_time[i]-burst_time[i];
            }
            printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, 
burst_time[i], sum-arrival_time[i], wt);
            tat = tat+sum-arrival_time[i];
            count =0;
        }
        if(i==n-1)
        {
            i=0;
        }
        else if(arrival_time[i+1]<=sum)
        {
            i++;
        }
        else
        {
            i=0;
        }
    }
    avg_wt = wt * 1.0/n;
    avg_tat = tat * 1.0/n;
    printf("\nAverage Turn Around Time: \t%f", avg_wt);
    printf("\nAverage Waiting Time: \t%f", avg_tat);
}
```

Example:



```
 Process No             Burst Time              Turn Around Time                Waiting Time
Process No[2]           5                               9                       4
Process No[3]           3                               11                      12
Process No[4]           4                               14                      22
Process No[1]           9                               21                      34
Average Turn Around Time:       8.500000
Average Waiting Time:   13.750000
```