#include <stdio.h>

#include <stdlib.h>

#define MAX 10

typedef struct

{

int pid;

int burst\_time; /

int waiting\_time;

int turnaround\_time;

} Process;

void calculateWaitingTime(Process processes[], int n)

{

processes[0].waiting\_time = 0;

int i;

for (i = 1; i < n; i++)

{

processes[i].waiting\_time = processes[i-1].waiting\_time + processes[i-1].burst\_time;

}

}

void calculateTurnaroundTime(Process processes[], int n)

{

int i;

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

{

processes[i].turnaround\_time = processes[i].waiting\_time +processes[i].burst\_time;

}

}

void printProcesses(Process processes[], int n)

{

int i;

printf("PID\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

{

printf("%d\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst\_time,processes[i].waiting\_time, processes[i].turnaround\_time);

}

}

void roundRobinScheduling(Process processes[], int n, int quantum)

{

int remaining\_time[n];int i;

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

remaining\_time[i] = processes[i].burst\_time;

int t = 0;

while (1)

{

int done = 1;

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

{

if (remaining\_time[i] > 0)

{

done = 0;

if (remaining\_time[i] > quantum)

{

t += quantum;

remaining\_time[i] -= quantum;

}

else

{

t += remaining\_time[i];

processes[i].waiting\_time = t - processes[i].burst\_time;

remaining\_time[i] = 0;

}

}

}

if (done == 1)

break;

}

}

void feedbackQueueScheduling(Process processes[], int n, int quantum[], int num\_levels)

{

int i;

for (i = 0; i < num\_levels; i++)

{

printf("\nRunning processes in queue level %d with quantum %d:\n", i+1,quantum[i]);

roundRobinScheduling(processes, n, quantum[i]);

calculateTurnaroundTime(processes, n);

printProcesses(processes, n);

}

}

int main()

{

int n, num\_levels;int i;

printf("Enter the number of processes: ");

scanf("%d", &n);

Process processes[MAX];

printf("Enter burst time for all process : ");

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

{

processes[i].pid = i+1;

scanf("%d", &processes[i].burst\_time);

}

printf("Enter the number of queue levels: ");

scanf("%d", &num\_levels);

int quantum[num\_levels];

printf("Enter time quantum for all levels : ");

for (i = 0; i < num\_levels; i++)

{

scanf("%d", &quantum[i]);

}

feedbackQueueScheduling(processes, n, quantum, num\_levels);

return 0;

}

OUTPUT

Enter the number of processes: 3

Enter burst time for all process : 5

3

8

Enter the number of queue levels: 2

Enter time quantum for all levels : 4 6

Running processes in queue level 1 with quantum 4:

PID Burst Time Waiting Time Turnaround Time

1 5 7 12

2 3 4 7

3 8 8 16

Running processes in queue level 2 with quantum 6:

PID Burst Time Waiting Time Turnaround Time

1 5 0 5

2 3 5 8

3 8 8 16