**Mutltilevel Queue Scheduling**

**Program:**

#include <stdio.h>

#include <stdlib.h>

typedef struct {

int id;

int at, bt, ct, wt, tat, queue;

} process;

void fcfs(process p[], int n, process final\_order[], int \*index);

void roundRobin(process p[], int n, int q, process final\_order[], int \*index);

void display(process p[], int n);

int total\_wt = 0, total\_tat = 0;

int main() {

int n, q, user\_count = 0, s\_count = 0;

printf("Enter number of processes: ");

scanf("%d", &n);

process p[n], user[n], system[n], final\_order[n];

int final\_index = 0;

printf("To differentiate between types of queues: \nEnter 0 for system process, 1 for user process\n");

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

printf("Enter Process ID, Arrival Time, Burst Time, and Type of queue for Process %d: ", i + 1);

scanf("%d %d %d %d", &p[i].id, &p[i].at, &p[i].bt, &p[i].queue);

}

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

if (p[i].queue == 0) {

system[s\_count++] = p[i];

} else {

user[user\_count++] = p[i];

}

}

if (s\_count > 0) {

printf("Enter time quantum for Round Robin: ");

scanf("%d", &q);

roundRobin(system, s\_count, q, final\_order, &final\_index);

}

if (user\_count > 0) {

fcfs(user, user\_count, final\_order, &final\_index);

}

display(final\_order, final\_index);

return 0;

}

void fcfs(process p[], int n, process final\_order[], int \*index) {

int currentTime = 0;

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

if (currentTime < p[i].at) {

currentTime = p[i].at;

}

p[i].ct = currentTime + p[i].bt;

p[i].tat = p[i].ct - p[i].at;

p[i].wt = p[i].tat - p[i].bt;

total\_wt += p[i].wt;

total\_tat += p[i].tat;

currentTime = p[i].ct;

final\_order[\*index] = p[i];

final\_order[\*index].queue = 1;

(\*index)++;

}

}

void roundRobin(process p[], int n, int q, process final\_order[], int \*index) {

int currentTime = 0, completed = 0;

int remaining\_time[n];

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

remaining\_time[i] = p[i].bt;

}

while (completed < n) {

int all\_idle = 1;

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

if (p[i].at <= currentTime && remaining\_time[i] > 0) {

all\_idle = 0;

int time\_slice = (remaining\_time[i] > q) ? q : remaining\_time[i];

currentTime += time\_slice;

remaining\_time[i] -= time\_slice;

if (remaining\_time[i] == 0) {

completed ++;

p[i].ct = currentTime;

p[i].tat = p[i].ct - p[i].at;

p[i].wt = p[i].tat - p[i].bt;

total\_wt += p[i].wt;

total\_tat += p[i].tat;

final\_order[\*index] = p[i];

final\_order[\*index].queue = 0;

(\*index)++;

}

}

}

if (all\_idle && completed < n) {

currentTime++;

}

}

}

void display(process p[], int n) {

printf("\nFinal Scheduling Table:\n");

printf("PID\tArrival\tBurst\tCompletion\tWaiting\tTurnaround\tQueue Type\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\t%s\n",

p[i].id, p[i].at, p[i].bt, p[i].ct, p[i].wt, p[i].tat,

(p[i].queue == 0) ? "RR" : "FCFS");

}

printf("\nAverage Waiting Time: %.2f\n", (float)total\_wt / n);

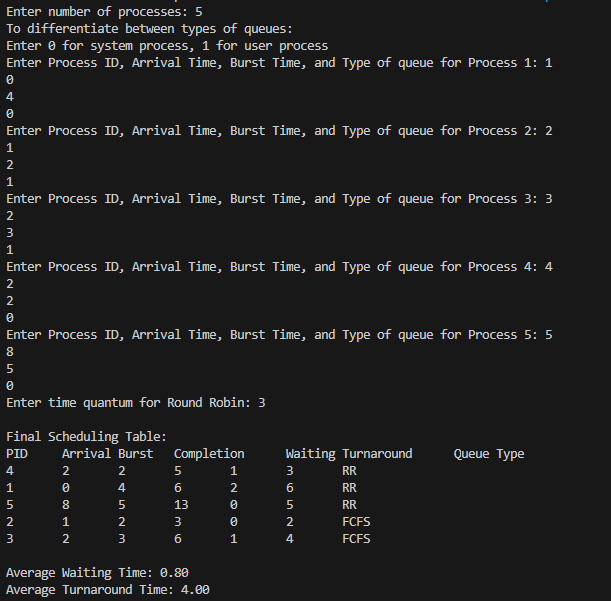
printf("Average Turnaround Time: %.2f\n", (float)total\_tat / n);

}

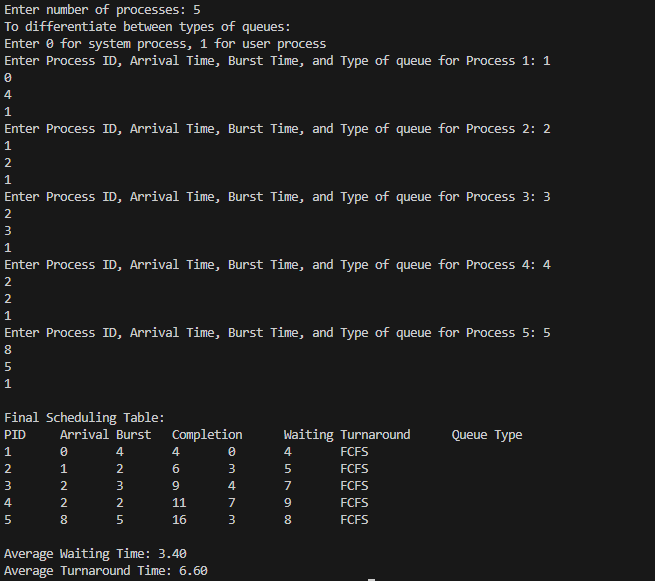
**OUTPUT:**

1. System queue: Round Robin

User Queue: FCFS



1. System queue and User queue : FCFS



1. System queue and User queue : Round Robin

