**4ITRC2 Operating System Lab**

**Lab Assignment 5**

**Aim**: To create C programs for the different scheduling algorithms.

**To perform:** Create and execute C programs for following CPU Scheduling Algorithms:

1. First Come First Serve (FCFS)

2. Shortest Job First (SJF)

3. Round Robin Scheduling

**To Submit:** C Codes for the above scheduling algorithms with their outputs.

**First Come First Serve (FCFS) Scheduling**

**Code-**

#include <stdio.h>

void FCFS(int processes[], int n, int bt[]) {

int wt[n], tat[n];

wt[0] = 0;

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

wt[i] = bt[i-1] + wt[i-1];

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

tat[i] = bt[i] + wt[i];

printf("\nFCFS Scheduling\nProcess\tBT\tWT\tTAT\n");

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

printf("P%d\t%d\t%d\t%d\n", processes[i], bt[i], wt[i], tat[i]);

}

int main() {

int processes[] = {1, 2, 3};

int n = sizeof processes / sizeof processes[0];

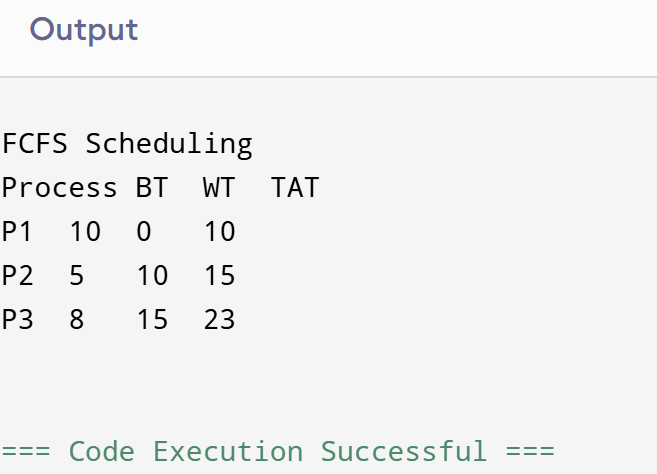
int bt[] = {10, 5, 8};

FCFS(processes, n, bt);

return 0;

}

**Output-**

****

**Shortest Job First (SJF) Scheduling**

**Code-**

#include <stdio.h>

void SJF(int processes[], int n, int bt[]) {

int wt[n], tat[n], temp, i, j;

// Sort by burst time (ascending)

for (i = 0; i < n - 1; i++) {

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

if (bt[i] > bt[j]) {

temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;

temp = processes[i]; processes[i] = processes[j]; processes[j] = temp;

}

}

}

wt[0] = 0;

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

wt[i] = bt[i-1] + wt[i-1];

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

tat[i] = bt[i] + wt[i];

printf("\nSJF Scheduling\nProcess\tBT\tWT\tTAT\n");

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

printf("P%d\t%d\t%d\t%d\n", processes[i], bt[i], wt[i], tat[i]);

}

int main() {

int processes[] = {1, 2, 3};

int n = sizeof processes / sizeof processes[0];

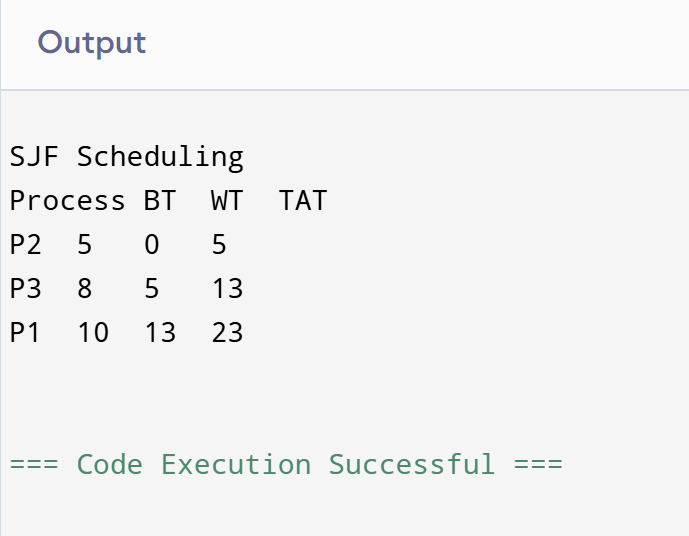
int bt[] = {10, 5, 8};

SJF(processes, n, bt);

return 0;

}

**Ouput-**

****

**Round Robin Scheduling**

**Code-**

#include <stdio.h>

void RoundRobin(int processes[], int n, int bt[], int quantum) {

int rem\_bt[n], wt[n], tat[n];

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

rem\_bt[i] = bt[i];

int t = 0;

while (1) {

int done = 1;

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

if (rem\_bt[i] > 0) {

done = 0;

if (rem\_bt[i] > quantum) {

t += quantum;

rem\_bt[i] -= quantum;

} else {

t += rem\_bt[i];

wt[i] = t - bt[i];

rem\_bt[i] = 0;

}

}

}

if (done == 1)

break;

}

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

tat[i] = bt[i] + wt[i];

printf("\nRound Robin Scheduling\nProcess\tBT\tWT\tTAT\n");

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

printf("P%d\t%d\t%d\t%d\n", processes[i], bt[i], wt[i], tat[i]);

}

int main() {

int processes[] = {1, 2, 3};

int n = sizeof processes / sizeof processes[0];

int bt[] = {10, 5, 8};

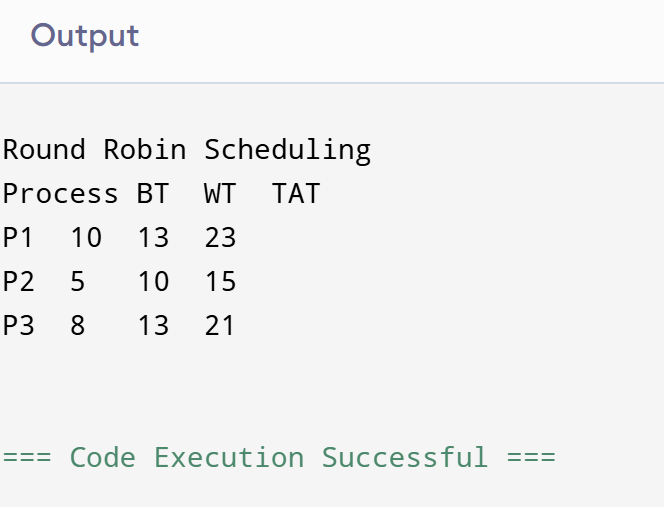
int quantum = 2;

RoundRobin(processes, n, bt, quantum);

return 0;

}

**Output-**

****