Name – Krishnaprasad Awala

Enrollment no – MITU21BTITD006

Roll no – D2215004

Lab 2

Shortest Job First :

Code :

#include <iostream>

#include <algorithm>

using namespace std;

struct Process {

int pid;

int arrivalTime;

int burstTime;

int completionTime;

int waitingTime;

int turnaroundTime;

};

bool compareArrivalTime(Process p1, Process p2) {

return p1.arrivalTime < p2.arrivalTime;

}

bool compareBurstTime(Process p1, Process p2) {

return p1.burstTime < p2.burstTime;

}

int main() {

int n, i;

float avwt = 0, avtat = 0;

cout << "Enter total number of processes: ";

cin >> n;

Process processes[n];

cout << "\nEnter Arrival Time and Burst Time for each process:\n";

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

cout << "P[" << i + 1 << "]: ";

processes[i].pid = i + 1;

cin >> processes[i].arrivalTime >> processes[i].burstTime;

}

// sort processes based on their arrival time

sort(processes, processes + n, compareArrivalTime);

int currentTime = processes[0].arrivalTime;

// execute processes in order of shortest burst time

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

int j;

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

if (processes[j].arrivalTime > currentTime) {

break;

}

}

sort(processes + i, processes + j, compareBurstTime);

processes[i].completionTime = currentTime + processes[i].burstTime;

processes[i].turnaroundTime = processes[i].completionTime - processes[i].arrivalTime;

processes[i].waitingTime = processes[i].turnaroundTime - processes[i].burstTime;

currentTime = processes[i].completionTime;

}

cout << "\nProcess\tArrival Time\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround Time";

// display process details and calculate average waiting and turnaround time

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

avwt += processes[i].waitingTime;

avtat += processes[i].turnaroundTime;

cout << "\nP[" << processes[i].pid << "]\t" << processes[i].arrivalTime << "\t\t" << processes[i].burstTime << "\t\t" << processes[i].completionTime << "\t\t" << processes[i].waitingTime << "\t\t" << processes[i].turnaroundTime;

}

avwt /= n;

avtat /= n;

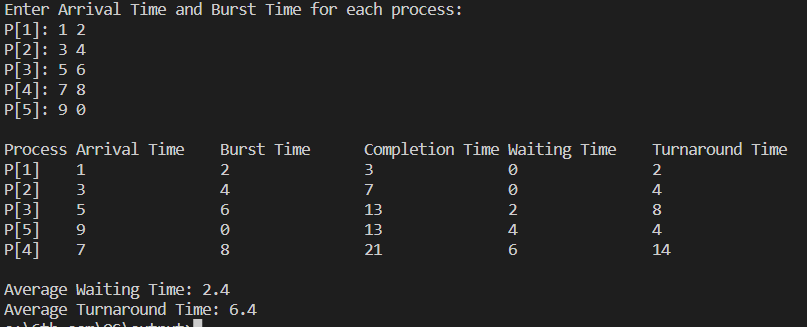
cout << "\n\nAverage Waiting Time: " << avwt;

cout << "\nAverage Turnaround Time: " << avtat;

return 0;

}

OutPut :



FCFS :

Code :

#include <iostream>

using namespace std;

int main() {

int n, at[20], bt[20], wt[20], tat[20], ct[20], i, j;

float avwt = 0, avtat = 0;

cout << "Enter total number of processes: ";

cin >> n;

cout << "\nEnter Arrival Time and Burst Time for each process:\n";

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

cout << "P[" << i + 1 << "]: ";

cin >> at[i] >> bt[i];

}

// bubble sort to sort processes based on their arrival time

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

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

if (at[j] > at[j + 1]) {

swap(at[j], at[j + 1]);

swap(bt[j], bt[j + 1]);

}

}

}

ct[0] = bt[0];

// calculating completion time for each process

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

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

}

wt[0] = 0;

// waiting time for first process is 0

// calculating waiting time

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

wt[i] = 0;

for (j = 0; j < i; j++) {

wt[i] += bt[j];

}

wt[i] -= at[i];

if (wt[i] < 0) {

wt[i] = 0;

}

}

cout << "\nProcess\tArrival Time\tBurst Time\tCompletion Time\tWaiting Time\tTurnaround Time";

// calculating turnaround time

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

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

avwt += wt[i];

avtat += tat[i];

cout << "\nP[" << i + 1 << "]\t" << at[i] << "\t\t" << bt[i] << "\t\t" << ct[i] << "\t\t" << wt[i] << "\t\t" << tat[i];

}

avwt /= i;

avtat /= i;

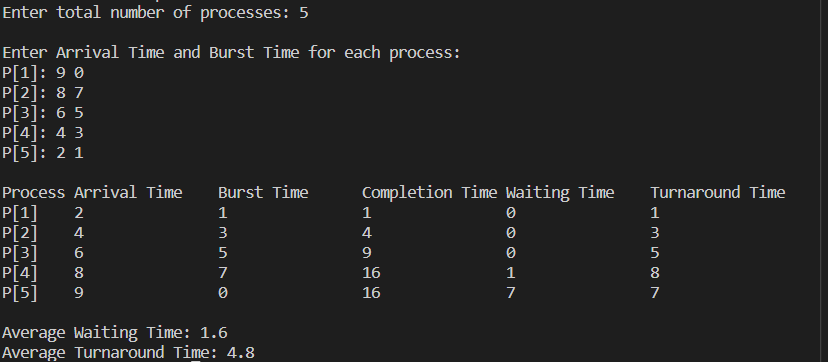
cout << "\n\nAverage Waiting Time: " << avwt;

cout << "\nAverage Turnaround Time: " << avtat;

return 0;

}

Output :



Round Robin :

Code :

#include <iostream>

#include <vector>

using namespace std;

struct Process {

int id;

int arrival\_time;

int burst\_time;

int remaining\_time;

int completion\_time;

int turnaround\_time;

int waiting\_time;

};

int main() {

int n, time\_quantum;

cout << "Enter the number of processes: ";

cin >> n;

cout << "Enter the time quantum: ";

cin >> time\_quantum;

vector<Process> processes(n);

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

cout << "Enter arrival time and burst time for process " << i+1 << ": ";

cin >> processes[i].arrival\_time >> processes[i].burst\_time;

processes[i].id = i+1;

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

}

int current\_time = 0;

int total\_waiting\_time = 0;

int total\_turnaround\_time = 0;

bool all\_processes\_completed;

do {

all\_processes\_completed = true;

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

Process& current\_process = processes[i];

if (current\_process.remaining\_time > 0) {

all\_processes\_completed = false;

int executed\_time = min(time\_quantum, current\_process.remaining\_time);

current\_process.remaining\_time -= executed\_time;

current\_time += executed\_time;

if (current\_process.remaining\_time == 0) {

current\_process.completion\_time = current\_time;

current\_process.turnaround\_time = current\_process.completion\_time - current\_process.arrival\_time;

current\_process.waiting\_time = current\_process.turnaround\_time - current\_process.burst\_time;

total\_waiting\_time += current\_process.waiting\_time;

total\_turnaround\_time += current\_process.turnaround\_time;

}

}

}

} while (!all\_processes\_completed);

double avg\_waiting\_time = static\_cast<double>(total\_waiting\_time) / n;

double avg\_turnaround\_time = static\_cast<double>(total\_turnaround\_time) / n;

cout << "Process\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n";

for (Process p : processes) {

cout << p.id << "\t\t" << p.arrival\_time << "\t\t" << p.burst\_time << "\t\t";

if (p.completion\_time > 0) {

cout << p.completion\_time << "\t\t";

cout << p.turnaround\_time << "\t\t\t" << p.waiting\_time << endl;

} else {

cout << "Not Completed\n";

}

}

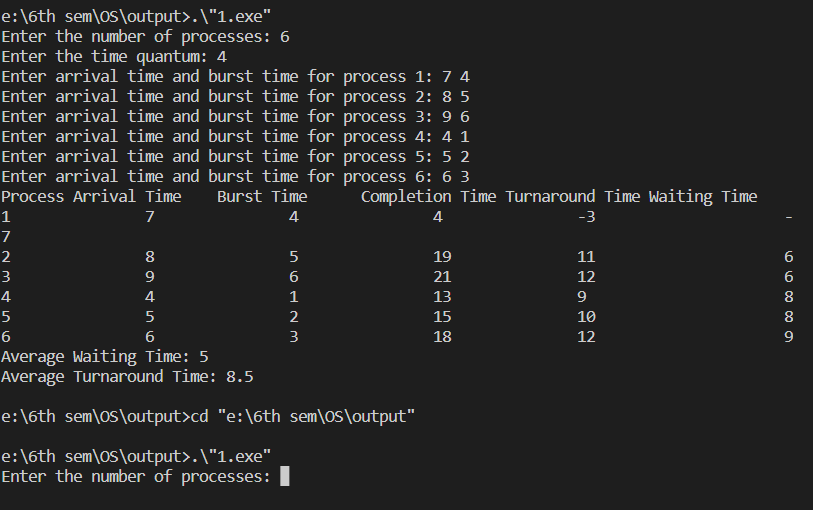
cout << "Average Waiting Time: " << avg\_waiting\_time << endl;

cout << "Average Turnaround Time: " << avg\_turnaround\_time << endl;

return 0;

}

Output :



Priority :

Code :

#include <iostream>

using namespace std;

int main() {

int n, at[20], bt[20], wt[20], tat[20], ct[20], p[20], pri[20], i, j;

float avwt = 0, avtat = 0;

cout << "Enter total number of processes: ";

cin >> n;

cout << "\nEnter Arrival Time, Burst Time and Priority for each process:\n";

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

cout << "P[" << i + 1 << "]: ";

cin >> at[i] >> bt[i] >> pri[i];

p[i] = i + 1;

}

// sorting the processes according to arrival time

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

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

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

swap(at[i], at[j]);

swap(bt[i], bt[j]);

swap(pri[i], pri[j]);

swap(p[i], p[j]);

}

}

}

ct[0] = bt[0];

// calculating completion time for each process

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

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

}

wt[0] = 0;

// waiting time for first process is 0

// calculating waiting time

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

wt[i] = 0;

for (j = 0; j < i; j++) {

wt[i] += bt[j];

}

wt[i] -= at[i];

if (wt[i] < 0) {

wt[i] = 0;

}

}

cout << "\nProcess\tArrival Time\tBurst Time\tPriority\tCompletion Time\tWaiting Time\tTurnaround Time";

// calculating turnaround time

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

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

avwt += wt[i];

avtat += tat[i];

cout << "\nP[" << p[i] << "]\t" << at[i] << "\t\t" << bt[i] << "\t\t" << pri[i] << "\t\t" << ct[i] << "\t\t" << wt[i] << "\t\t" << tat[i];

}

avwt /= i;

avtat /= i;

cout << "\n\nAverage Waiting Time: " << avwt;

cout << "\nAverage Turnaround Time: " << avtat;

return 0;

}

Output :

