Experiment Number: 7

Problem Statement: **CPU Scheduling Algorithms**

Name: Arnav Shah Roll No. : 21

Class : AI\_C Batch : B2

1. **First Come First Search :-**

#include<iostream>

using namespace std;

// Function to find the waiting time for all processes

void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[]) {

int service\_time[n];

service\_time[0] = at[0]; // Service time for first process is its arrival time

wt[0] = 0; // Waiting time for first process is 0

// calculating waiting time

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

// Calculating service time for each process

service\_time[i] = service\_time[i - 1] + bt[i - 1];

// If the current process hasn't arrived yet, wait until it arrives

if (service\_time[i] < at[i])

service\_time[i] = at[i];

// Calculate waiting time

wt[i] = service\_time[i] - at[i];

// If waiting time is negative, make it 0

if (wt[i] < 0)

wt[i] = 0;

}

}

// Function to calculate turn around time

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {

// calculating turnaround time by adding bt[i] + wt[i]

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

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

}

// Function to calculate average time

void findavgTime(int processes[], int n, int bt[], int at[]) {

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

// Function to find waiting time of all processes

findWaitingTime(processes, n, bt, wt, at);

// Function to find turn around time for all processes

findTurnAroundTime(processes, n, bt, wt, tat);

// Display processes along with all details

cout << "Processes " << " Arrival time " << " Burst time "

<< " Waiting time " << " Turn around time\n";

// Calculate total waiting time and total turn around time

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

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

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

cout << " " << processes[i] << "\t\t" << at[i] << "\t\t"

<< bt[i] << "\t " << wt[i] << "\t\t " << tat[i] << endl;

}

cout << "Average waiting time = "

<< (float)total\_wt / (float)n;

cout << "\nAverage turn around time = "

<< (float)total\_tat / (float)n;

}

int main() {

int n;

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

cin >> n;

int processes[n];

int arrival\_time[n];

int burst\_time[n];

cout << "Enter arrival time for each process:\n";

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

cout << "Arrival time of process " << i + 1 << ": ";

cin >> arrival\_time[i];

processes[i] = i + 1; // Assigning process IDs

}

cout << "Enter burst time for each process:\n";

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

cout << "Burst time for process " << i + 1 << ": ";

cin >> burst\_time[i];

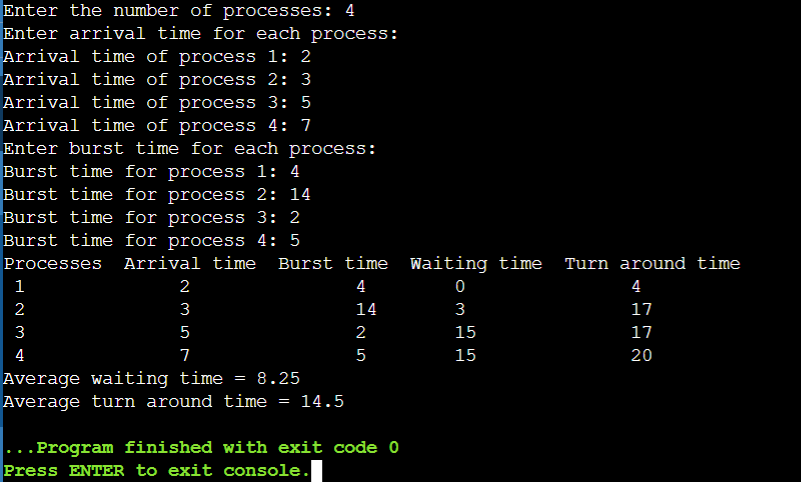
}

findavgTime(processes, n, burst\_time, arrival\_time);

return 0;

}

**Output :-**



1. **Shortest Job First :-**

struct Process {

int arrival\_time;

int burst\_time;

int waiting\_time;

};

int compare(const void \*a, const void \*b) {

struct Process \*p1 = (struct Process \*)a;

struct Process \*p2 = (struct Process \*)b;

return p1->burst\_time - p2->burst\_time;

}

int main() {

int n, i, j;

float avg\_waiting\_time = 0, avg\_turnaround\_time = 0;

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

scanf("%d", &n);

struct Process processes[n];

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

printf("Enter arrival time and burst time of process %d: ", i+1);

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

}

qsort(processes, n, sizeof(struct Process), compare);

processes[0].waiting\_time = 0;

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

processes[i].waiting\_time = 0;

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

{

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

}

avg\_waiting\_time += processes[i].waiting\_time;

}

avg\_waiting\_time /= n;

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

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

}

avg\_turnaround\_time /= n;

printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i+1, processes[i].arrival\_time, processes[i].burst\_time, processes[i].waiting\_time, processes[i].burst\_time+processes[i].waiting\_time);

}

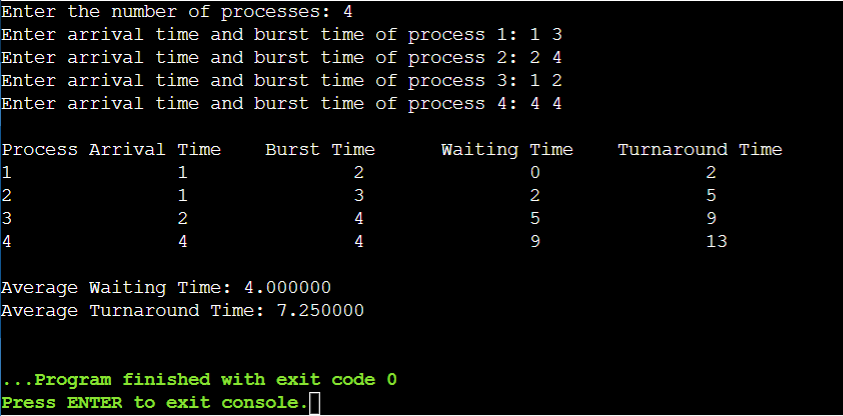
printf("\nAverage Waiting Time: %f\n", avg\_waiting\_time);

printf("Average Turnaround Time: %f\n", avg\_turnaround\_time);

return 0;

}

**Output :-**



1. **Round Robin :-**

#include <iostream>

#include <climits>

using namespace std;

struct Process {

int AT, BT, ST[20], WT, FT, TAT, pos;

};

int quant;

int main() {

int n, i, j;

// Taking Input

cout << "Enter the no. of processes: ";

cin >> n;

Process p[n];

cout << "Enter the quantum: " << endl;

cin >> quant;

cout << "Enter the process numbers: " << endl;

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

cin >> p[i].pos;

cout << "Enter the Arrival time of processes: " << endl;

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

cin >> p[i].AT;

cout << "Enter the Burst time of processes: " << endl;

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

cin >> p[i].BT;

// Declaring variables

int c = n, s[n][20];

float time = 0, mini = INT\_MAX, b[n], a[n];

// Initializing burst and arrival time arrays

int index = -1;

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

b[i] = p[i].BT;

a[i] = p[i].AT;

for (j = 0; j < 20; j++) {

s[i][j] = -1;

}

}

int tot\_wt, tot\_tat;

tot\_wt = 0;

tot\_tat = 0;

bool flag = false;

while (c != 0) {

mini = INT\_MAX;

flag = false;

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

float p = time + 0.1;

if (a[i] <= p && mini > a[i] && b[i] > 0) {

index = i;

mini = a[i];

flag = true;

}

}

// if at =1 then loop gets out hence set flag to false

if (!flag) {

time++;

continue;

}

// calculating start time

j = 0;

while (s[index][j] != -1) {

j++;

}

if (s[index][j] == -1) {

s[index][j] = time;

p[index].ST[j] = time;

}

if (b[index] <= quant) {

time += b[index];

b[index] = 0;

} else {

time += quant;

b[index] -= quant;

}

if (b[index] > 0) {

a[index] = time + 0.1;

}

// calculating arrival, burst, final times

if (b[index] == 0) {

c--;

p[index].FT = time;

p[index].WT = p[index].FT - p[index].AT - p[index].BT;

tot\_wt += p[index].WT;

p[index].TAT = p[index].BT + p[index].WT;

tot\_tat += p[index].TAT;

}

} // end of while loop

// Printing output

cout << "Process number ";

cout << "Arrival time ";

cout << "Burst time ";

cout << "\tStart time";

j = 0;

while (j != 10) {

j += 1;

cout << " ";

}

cout << "\t\tFinal time";

cout << "\tWait Time ";

cout << "\tTurnAround Time" << endl;

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

cout << p[i].pos << "\t\t";

cout << p[i].AT << "\t\t";

cout << p[i].BT << "\t";

j = 0;

int v = 0;

while (s[i][j] != -1) {

cout << p[i].ST[j] << " ";

j++;

v += 3;

}

while (v != 40) {

cout << " ";

v += 1;

}

cout << p[i].FT << "\t\t";

cout << p[i].WT << "\t\t";

cout << p[i].TAT << endl;

}

// Calculating average wait time and turnaround time

double avg\_wt, avg\_tat;

avg\_wt = tot\_wt / static\_cast<double>(n);

avg\_tat = tot\_tat / static\_cast<double>(n);

// Printing average wait time and turnaround time

cout << "The average wait time is: " << avg\_wt << endl;

cout << "The average TurnAround time is: " << avg\_tat << endl;

return 0;

}

**Output :-**

