Experiment No. 5

**AIM :** a) Write a program to demonstrate the concept of non-preemptive scheduling algorithm [FCFS].

# Program:

#include <stdio.h> #define MAX 10

// Structure to store process details struct Process {

int id;

int arrival\_time; int burst\_time;

int completion\_time; int turn\_around\_time; int waiting\_time;

};

// Function to calculate turnaround time and waiting time for FCFS void calculate\_times(struct Process proc[], int n) {

int total\_turnaround\_time = 0; int total\_waiting\_time = 0;

proc[0].completion\_time = proc[0].arrival\_time + proc[0].burst\_time; proc[0].turn\_around\_time = proc[0].completion\_time - proc[0].arrival\_time; proc[0].waiting\_time = proc[0].turn\_around\_time - proc[0].burst\_time;

total\_turnaround\_time += proc[0].turn\_around\_time; total\_waiting\_time += proc[0].waiting\_time;

// Calculate times for the rest of the processes for (int i = 1; i < n; i++) {

proc[i].completion\_time = proc[i-1].completion\_time + proc[i].burst\_time; proc[i].turn\_around\_time = proc[i].completion\_time - proc[i].arrival\_time; proc[i].waiting\_time = proc[i].turn\_around\_time - proc[i].burst\_time;

total\_turnaround\_time += proc[i].turn\_around\_time; total\_waiting\_time += proc[i].waiting\_time;

}

printf("Process ID\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n"); for (int i = 0; i < n; i++) {

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

proc[i].id, proc[i].arrival\_time, proc[i].burst\_time, proc[i].completion\_time, proc[i].turn\_around\_time, proc[i].waiting\_time);

}

printf("\nAverage Turnaround Time: %.2f\n", (float)total\_turnaround\_time / n); printf("Average Waiting Time: %.2f\n", (float)total\_waiting\_time / n);

}

int main() {

struct Process proc[MAX]; int n;

printf("Enter number of processes: "); scanf("%d", &n);

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

printf("Enter details for process %d:\n", i + 1); proc[i].id = i + 1;

printf("Arrival time: "); scanf("%d", &proc[i].arrival\_time); printf("Burst time: ");

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

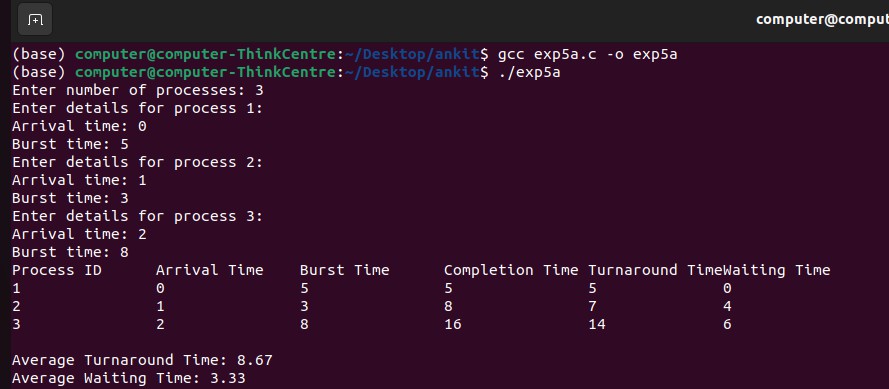
}

calculate\_times(proc, n);

return 0;

}

# Output:

****

**AIM :** b) Write a program to demonstrate the concept of non-preemptive scheduling algorithm [SRTN].

# Program:

#include <stdio.h> #define MAX 10

struct Process { int id;

int arrival\_time; int burst\_time;

int remaining\_time; int completion\_time; int turn\_around\_time; int waiting\_time;

};

int find\_shortest\_remaining\_time(struct Process proc[], int n, int current\_time) { int min\_time = 9999;

int min\_index = -1;

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

if (proc[i].arrival\_time <= current\_time && proc[i].remaining\_time > 0) { if (proc[i].remaining\_time < min\_time) {

min\_time = proc[i].remaining\_time; min\_index = i;

}

}

}

return min\_index;

}

void calculate\_times(struct Process proc[], int n) { int total\_turnaround\_time = 0;

int total\_waiting\_time = 0; int current\_time = 0;

int completed = 0;

while (completed < n) {

int idx = find\_shortest\_remaining\_time(proc, n, current\_time);

if (idx == -1) { current\_time++; continue;

}

proc[idx].remaining\_time--; current\_time++;

if (proc[idx].remaining\_time == 0) { proc[idx].completion\_time = current\_time;

proc[idx].turn\_around\_time = proc[idx].completion\_time - proc[idx].arrival\_time; proc[idx].waiting\_time = proc[idx].turn\_around\_time - proc[idx].burst\_time;

total\_turnaround\_time += proc[idx].turn\_around\_time; total\_waiting\_time += proc[idx].waiting\_time; completed++;

}

}

printf("Process ID\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n"); for (int i = 0; i < n; i++) {

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

proc[i].id, proc[i].arrival\_time, proc[i].burst\_time, proc[i].completion\_time, proc[i].turn\_around\_time, proc[i].waiting\_time);

}

printf("\nAverage Turnaround Time: %.2f\n", (float)total\_turnaround\_time / n); printf("Average Waiting Time: %.2f\n", (float)total\_waiting\_time / n);

}

int main() {

struct Process proc[MAX]; int n;

printf("Enter number of processes: "); scanf("%d", &n);

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

printf("Enter details for process %d:\n", i + 1); proc[i].id = i + 1;

printf("Arrival time: "); scanf("%d", &proc[i].arrival\_time); printf("Burst time: ");

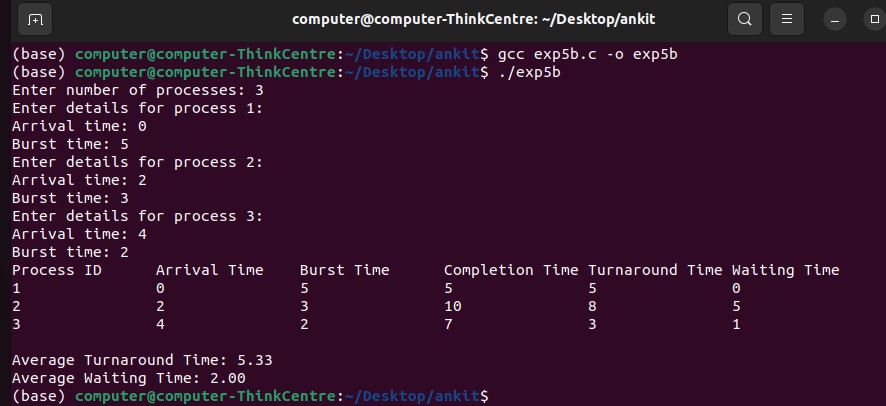
scanf("%d", &proc[i].burst\_time); proc[i].remaining\_time = proc[i].burst\_time;

}

calculate\_times(proc, n);

return 0;

}

**Output:**