EXPERIMENT NO: 2 **DATE:**

TITLE: Non–Pre-emptive CPU Scheduling Algorithm

AIM: To implement **First Come First Serve (FCFS)** Scheduling Algorithm.

THEORY

First Come First Served (FCFS) is a **non-pre-emptive** scheduling algorithm that follows the **FIFO (First In First Out)** strategy, assigning priority to processes in the order they request the processor.

- The process that requests the CPU first is allocated the CPU first.
- This is implemented using a **FIFO queue**:
    - As processes arrive, they are added to the end of the queue.
    - When the CPU becomes free, it takes the process from the start of the queue.

Given *n* processes with their burst times, the task is to calculate:

- **Average Waiting Time**
- **Average Turnaround Time**

In FCFS, the process that comes first is executed first, and the next process starts **only after** the previous one finishes.

Assumption: Arrival time for all processes is 0.

Advantages:

- Simple and easy to implement.

Disadvantages:

- **Starvation** may occur if the first process has a very large burst time.

Scheduling Purpose:
Scheduling ensures processes are completed on time by efficiently utilizing CPU and I/O. In multiprogramming systems, one process can use the CPU while another waits for I/O, improving overall performance.

TERMINOLOGIES

1. Arrival Time: Time at which the process arrives in the ready queue.

2. Completion Time: Time at which process completes its execution.

3. Burst Time: Time required by a process for CPU execution.

4. Turn Around Time: Time Difference between completion time and arrival time.

i) Turn Around Time = Completion Time – Arrival Time

5. Waiting Time(W.T): Time Difference between turnaround time and burst time.

i) Waiting Time = Turn Around Time – Burst Time

6. Min turnaround time: Time taken by a process to finish execution

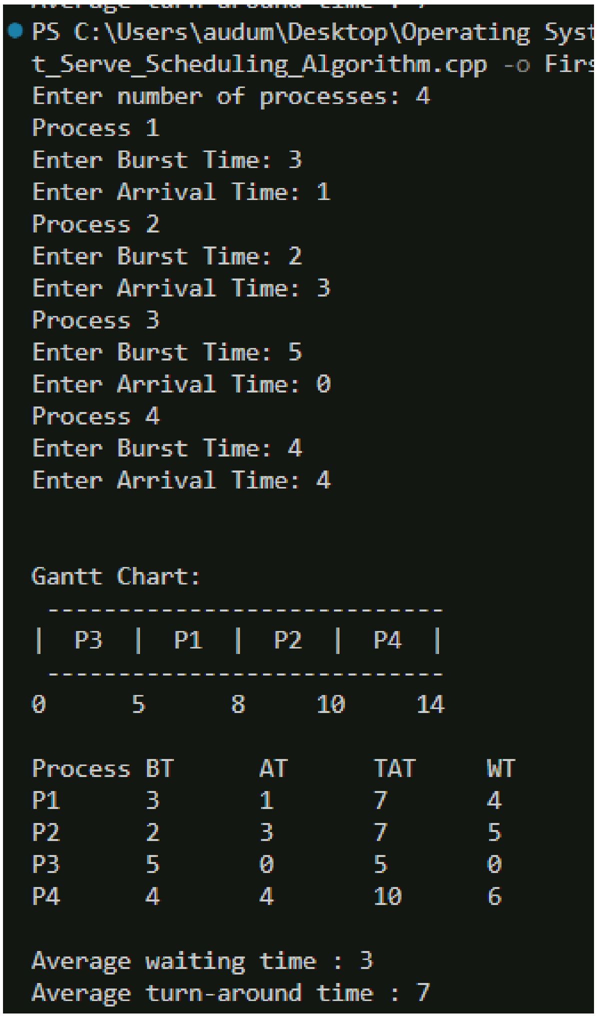
7. Min waiting time: Time a process waits in ready queue

8. Min response time: Time when a process produces first response

CODE

```
#include <iostream>
#include <iomanip>
using namespace std;

int order[10];

int average(int *matrix, int n) {
    int avg = 0;
    for (int i = 0; i < n; i++)
        avg += matrix[i];
    return avg / n;
}

void sort_arr(int arr[], int n) {
    int copy[n];
    for (int i = 0; i < n; i++) {
        order[i] = i;
        copy[i] = arr[i];
    }
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (copy[i] > copy[j]) {
                int temp = copy[i];
                copy[i] = copy[j];
                copy[j] = temp;

                int t = order[i];
                order[i] = order[j];
                order[j] = t;
            }
        }
    }
}

void calc(int burst[], int arr[], int n) {
    sort_arr(arr, n);
    int complete[n], TAT[n], WT[n], time = 0;

    for (int i = 0; i < n; i++) {
        time += burst[order[i]];
        complete[order[i]] = time; }
    for (int i = 0; i < n; i++)
        TAT[i] = complete[i] - arr[i];
    for (int i = 0; i < n; i++)
        WT[i] = TAT[i] - burst[i];
    cout << "\nGantt Chart:\n";
    cout << " ";
    for (int i = 0; i < n; i++)
        cout << "-------";
    cout << "\n|";
    for (int i = 0; i < n; i++)
        cout << "  P" << order[i] + 1 << "  |";
    cout << "\n ";
    for (int i = 0; i < n; i++)
        cout << "-------";
    cout << "\n0";
    for (int i = 0; i < n; i++)
        cout << setw(7) << complete[order[i]];
    cout << "\n\n";
    cout << "Process\tBT\tAT\tTAT\tWT\n";
    for (int i = 0; i < n; i++) {
        cout << "P" << i + 1 << "\t"
             << burst[i] << "\t"
             << arr[i] << "\t"
             << TAT[i] << "\t"
             << WT[i] << endl; }
    cout << "\nAverage waiting time : " << average(WT, n);
    cout << "\nAverage turn-around time : " << average(TAT, n);
}
int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    int burst[n], arr[n];
    for (int i = 0; i < n; i++) {
        cout << "Process " << i + 1 << endl;
        cout << "Enter Burst Time: ";
        cin >> burst[i];
        cout << "Enter Arrival Time: ";
        cin >> arr[i];
    }

    cout << endl;
    calc(burst, arr, n);
    return 0;
}
```

OUPUT:



```
Average turn-around time : 7
PS C:\Users\audum\Desktop\Operating Syst
t_Serve_Scheduling_Algorithm.cpp -o Firs
Enter number of processes: 4
Process 1
Enter Burst Time: 3
Enter Arrival Time: 1
Process 2
Enter Burst Time: 2
Enter Arrival Time: 3
Process 3
Enter Burst Time: 5
Enter Arrival Time: 0
Process 4
Enter Burst Time: 4
Enter Arrival Time: 4


Gantt Chart:
 ----------------------------
|  P3  |  P1  |  P2  |  P4  |
 ----------------------------
0      5      8      10     14

Process BT      AT      TAT     WT
P1      3       1       7       4
P2      2       3       7       5
P3      5       0       5       0
P4      4       4       10      6

Average waiting time : 3
Average turn-around time : 7
```

Conclusion:first come first serve cpu scheduling algorithm was implemented successfully in cpp.