Experiment No – 3 c **Date -**

(c) NON-PREEMPTIVE PRIORITY SCHEDULING

Aim - To implement Non-Preemptive priority scheduling algorithm

THEORY –

Priority Scheduling is a CPU scheduling algorithm where each process is assigned a priority number.

In Non-Preemptive Priority Scheduling, once a process starts execution, it runs till completion — it cannot be interrupted, even if another process with a higher priority arrives later.

When the CPU is free, the process with the highest priority among the waiting processes is selected to execute.

CHARACTERISTICS

Non-preemptive: Once a process gets the CPU, it will not release it until completion.

Efficient for batch systems, but not suitable for interactive systems where urgent processes may arrive later.

Starvation problem: Low-priority processes may wait for a long time if high-priority processes keep arriving.

PRIORITY MECHANISM

Each process has a priority number.
Lower number = Higher priority (in most systems).
Example: Priority 1 is higher than Priority 3.
CPU is allocated to the waiting process with the highest priority.
If two processes have the same priority, then First-Come-First-Serve (FCFS) is used as a tie-breaker.

IMPORTANT TERMS

1. Arrival Time (AT)
The time at which a process arrives in the ready queue.
Example: If Process P1 has AT = 2, it means it comes to the system at time unit 2.

2. Burst Time (BT)

The total execution time a process needs to complete on the CPU.
Example: If BT = 5, the process requires 5 units of CPU time.

3. Completion Time (CT)
The time at which a process finishes execution.
Example: If a process starts at time 4 and runs for 5 units, its CT = 9.

4. Turnaround Time (TAT)
The total time a process spends in the system (from arrival to completion).
Formula:
TAT = CT - AT

5. Waiting Time (WT)
The time a process spends waiting in the ready queue before getting CPU.
Formula:
WT = TAT - BT

6. Average Turnaround Time (Avg TAT)
Average of turnaround times of all processes.

7. Average Waiting Time (Avg WT)
Average of waiting times of all processes.

Example -

Process Name	Arrival Time	Burst Time	Priority
A	2	2	3
B	0	8	4
C	3	16	1
D	7	1	2

Gantt Chart –

B	C	D	A

0 8 24 26 27

Table -

Process Name	Arrival Time	Burst Time	Priority	Completion Time	Turn Around Time (TAT)	Wait Time
A	2	2	3	27	25	23
B	0	8	4	8	8	0
C	3	16	1	24	21	5
D	7	1	2	26	19	18

AVERAGE TURN AROUND TIME IS 18.25 AND AVERAGE WAIT TIME IS 11.5

PROGRAM

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

struct Process {
    string name;
    int at, bt, ct, tat, wt, pr;
    bool done;
};

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    vector<Process> p(n);
    for (int i = 0; i < n; i++) {
        cout << "Enter Process Name, Arrival Time, Burst Time, Priority for P" << i + 1 << ": ";
        cin >> p[i].name >> p[i].at >> p[i].bt >> p[i].pr;
        p[i].done = false;
    }

    int time = 0, completed = 0;
    vector<string> ganttProcess;
    vector<int> ganttTime;
    ganttTime.push_back(0);

    while (completed != n) {
        int idx = -1;
        int highestPriority = 1e9;
        for (int i = 0; i < n; i++) {
            if (p[i].at <= time && !p[i].done) {
                if (p[i].pr < highestPriority) {
                    highestPriority = p[i].pr;
                    idx = i;
                } else if (p[i].pr == highestPriority) {
                    if (p[i].at < p[idx].at)
                        idx = i;
                }
            }
        }

        if (idx != -1) {
            time = max(time, p[idx].at);
            p[idx].ct = time + p[idx].bt;
            time = p[idx].ct;
            p[idx].tat = p[idx].ct - p[idx].at;
            p[idx].wt = p[idx].tat - p[idx].bt;
            p[idx].done = true;
            completed++;

            ganttProcess.push_back(p[idx].name);
            ganttTime.push_back(time);
        } else {
            time++;
            if (ganttProcess.empty() || ganttProcess.back() != "IDLE") {
                ganttProcess.push_back("IDLE");
                ganttTime.push_back(time);
            } else {
                ganttTime.back() = time;
            }
        }
    }

    int totalTAT = 0, totalWT = 0;
    for (int i = 0; i < n; i++) {
        totalTAT += p[i].tat;
        totalWT += p[i].wt;
    }

    cout << "--------------------------------------------------------------\n";
    cout << "Process\tAT\tBT\tPR\tCT\tTAT\tWT\n";
    cout << "--------------------------------------------------------------\n";
    for (int i = 0; i < n; i++) {
        cout << p[i].name << "\t" << p[i].at << "\t" << p[i].bt << "\t"
             << p[i].pr << "\t" << p[i].ct << "\t" << p[i].tat << "\t" << p[i].wt << "\n";
    }
    cout << "--------------------------------------------------------------\n";

    cout << "Total Turnaround Time = " << totalTAT << "\n";
    cout << "Average Turnaround Time = " << (float)totalTAT / n << "\n";
    cout << "Average Waiting Time = " << (float)totalWT / n << "\n\n";
```

```
  cout << "Gantt
Chart:\n";
  cout << "-----------------
-------------------------------
---------------\n|";
  for (auto& proc :
ganttProcess) {
      cout << "   " <<
setw(2) << proc << " |";
  }
  cout << "\n--------------
-------------------------------
------------------\n";
  for (size_t i = 0; i <=
ganttProcess.size(); i++)
{
      cout << setw(7) <<
ganttTime[i];
  }
  cout << "\n";

  return 0;
}
```

OUTPUT –

```
Enter number of processes: 4
Enter Process Name, Arrival Time, Burst Time, Priority for P1: A
2
2
3
Enter Process Name, Arrival Time, Burst Time, Priority for P2: B
0
8
4
Enter Process Name, Arrival Time, Burst Time, Priority for P3: C
3
16
1
Enter Process Name, Arrival Time, Burst Time, Priority for P4: D
5
1
2
---------------------------------------------------------------
---------------------------------------------------------------
Process AT       BT       PR       CT       TAT      WT
---------------------------------------------------------------
A       2        2        3        27       25       23
B       0        8        4        8        8        0
C       3        16       1        24       21       5
D       5        1        2        25       20       19
---------------------------------------------------------------
Total Turnaround Time = 74
Average Turnaround Time = 18.5
Average Waiting Time = 11.75

Gantt Chart:
---------------------------------------------------------------
|      B |      C |      D |      A |
---------------------------------------------------------------
      0       8      24      25      27
```

CONCLUSION -This scheduling successfully demonstrates how processes with higher priority are executed first, ensuring important tasks get precedence. Yet, it suffers from the problem of starvation for lower-priority processes, unless techniques like aging are applied to balance fairness and efficiency.