**VAISHNAVI E 231901059**

**Ex. No.: 6C**

**Date:19.03.2025**

**PRIORITY SCHEDULING**

**Aim:**

To implement priority scheduling technique

**Algorithm:**

1. Get the number of processes from the user.
2. Read the process name, burst time and priority of process.
3. Sort based on burst time of all processes in ascending order based priority 4.

Calculate the total waiting time and total turnaround time for each process 5.

Display the process name & burst time for each process.

6. Display the total waiting time, average waiting time, turnaround time

**Program Code:** #include <stdio.h> void sortProcesses(int n, int id[], int bt[], int priority[]) { for (int i = 0; i < n - 1; i++) { for (int j = i + 1; j < n; j++) { if (priority[i] > priority[j]) { int temp; temp = priority[i]; priority[i] = priority[j]; priority[j] = temp; temp = bt[i]; bt[i] = bt[j]; bt[j] = temp; temp = id[i]; id[i] = id[j]; id[j] = temp;

}

}

}

} void calculateTimes(int n, int bt[], int wt[], int tat[]) {

wt[0] = 0;

for (int i = 1; i < n; i++) { wt[i] = wt[i - 1] + bt[i - 1];

}

for (int i = 0; i < n; i++) { tat[i] = wt[i] + bt[i];

}

} void display(int n, int id[], int bt[], int priority[], int wt[], int tat[]) {

int totalWT = 0, totalTAT = 0; printf("\nProcess\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n"); for (int i = 0; i < n; i++) { totalWT += wt[i]; totalTAT += tat[i]; printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", id[i], bt[i], priority[i], wt[i], tat[i]);

}

printf("\nTotal Waiting Time: %d", totalWT); printf("\nAverage Waiting Time: %.2f", (float)totalWT / n); printf("\nTotal Turnaround Time: %d", totalTAT); printf("\nAverage Turnaround Time: %.2f\n", (float)totalTAT / n);

} void preemptivePriorityScheduling(int n, int id[], int bt[], int priority[]) { int remainingTime[n], completed = 0, time = 0, minPriority, minIndex, wt[n], tat[n]; for (int i = 0; i < n; i++) remainingTime[i] = bt[i];

while (completed < n) { minPriority = 9999, minIndex = -1; for (int i = 0; i < n; i++) { if (remainingTime[i] > 0 && priority[i] < minPriority) { minPriority = priority[i]; minIndex = i;

}

}

if (minIndex == -1) break; remainingTime[minIndex]--; time++; if (remainingTime[minIndex] == 0) { completed++; tat[minIndex] = time; wt[minIndex] = tat[minIndex] - bt[minIndex];

}

}

display(n, id, bt, priority, wt, tat);

} int main() {

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

int id[n], bt[n], priority[n], wt[n], tat[n]; for (int i = 0; i < n; i++) { id[i] = i + 1; printf("Enter burst time and priority for process P%d: ", i + 1); scanf("%d %d", &bt[i], &priority[i]);

}

printf("Choose Scheduling Type:\n1. Non-Preemptive\n2. Preemptive\nEnter choice: "); scanf("%d", &choice); if (choice == 1) { sortProcesses(n, id, bt, priority); calculateTimes(n, bt, wt, tat); display(n, id, bt, priority, wt, tat);

} else if (choice == 2) { preemptivePriorityScheduling(n, id, bt, priority);

} else {

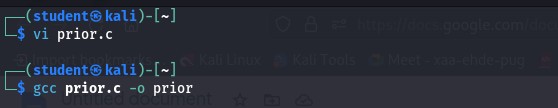
printf("Invalid choice!\n");

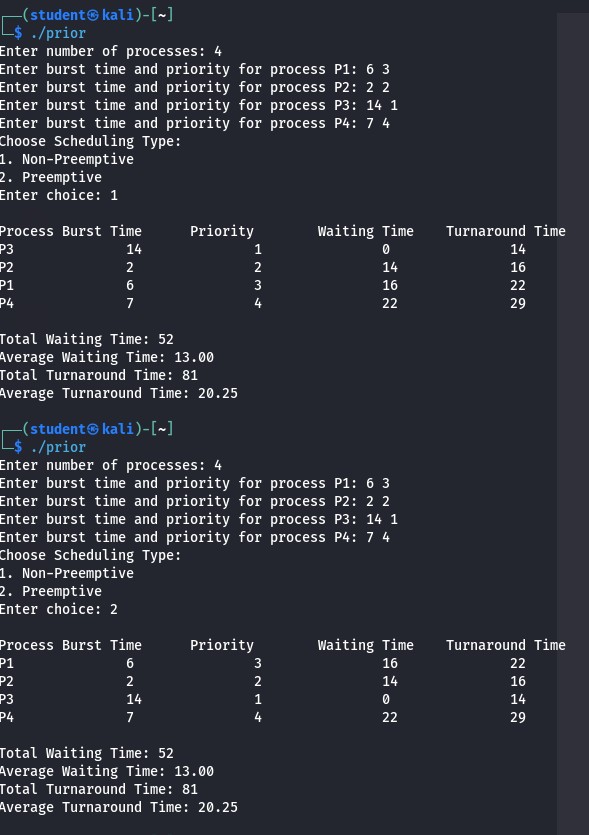
}

return 0;

}

**Output:**





**Result:**

Hence, priority scheduling in both preemptive and non-preemptive method has been executed successfully.