Write program to implement non-preemptive priority based scheduling algorithm

CODE

```cpp
//non-preemptive priority scheduling algorithm
#include <bits/stdc++.h>
#include <iostream>
using namespace std;

struct Process
{
    int pid;
    int bt;
    int priority;
};

bool comparison(Process a, Process b)
{
    return (a.priority > b.priority);
}

void findWaitingTime(Process proc[], int n, int wt[])
{
    wt[0] = 0;

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

void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
        tat[i] = proc[i].bt + wt[i];
}

void findavgTime(Process proc[], int n)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
```

```
    findWaitingTime(proc, n, wt);
    findTurnAroundTime(proc, n, wt, tat);

    cout << "\nProcesses  "
         << " CPU Burst time  "
         << " Waiting time  "
         << " Turn around time\n";

    for (int i = 0; i < n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << "    " << proc[i].pid << "\t\t" << proc[i].bt << "\t    " <<
wt[i] << "\t\t  " << tat[i] << endl;
    }

    cout << "\nAverage Waiting Time = " << (float)total_wt / (float)n;
    cout << "\nAverage Turn around Time = " << (float)total_tat / (float)n;
}

void priorityScheduling(Process proc[], int n)
{
    std::sort(proc, proc + n, comparison);

    cout << "\nOrder of execution: ";
    for (int i = 0; i < n; i++)
    {
        cout << proc[i].pid << " ";
    }
    cout<< endl;
    findavgTime(proc, n);
}

int main()
{
    int n;
    cout << "\nPriority Scheduling\nPlease enter the number of Processes = ";
    cin >> n;
    Process *proc = new Process[n];

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

```
    {
        cout << "\nPlease enter the CPU Burst Time for Process P" << i + 1 << "=
";
        cin >> proc[i].bt;
        cout << "Please enter the Priority of Process P" << i + 1 << "= ";
        cin >> proc[i].priority;
        proc[i].pid = i + 1;
    }
    priorityScheduling(proc, n);
    return 0;
}
```

OUTPUT

```
➜ OSPracticals g++ Practical9.cpp -o Practical9
➜ OSPracticals ./Practical9

Priority Scheduling
Please enter the number of Processes = 3

Please enter the CPU Burst Time for Process P1= 4
Please enter the Priority of Process P1= 2

Please enter the CPU Burst Time for Process P2= 6
Please enter the Priority of Process P2= 1

Please enter the CPU Burst Time for Process P3= 8
Please enter the Priority of Process P3= 4

Order of execution: 3 1 2

Processes   CPU Burst time   Waiting time   Turn around time
   3            8              0              8
   1            4              8              12
   2            6              12             18

Average Waiting Time = 6.66667
Average Turn around Time = 12.6667%
➜ OSPracticals
```

Write program to implement preemptive priority based scheduling algorithm

CODE

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

int main()
{
    int n;
    cout << "Please enter the number of processes: ";
    cin >> n;
    float total, wait[n];
    float p[n], twaiting = 0, waiting = 0;
    int proc;
    int stack[n];
    float brust[n], arrival[n], sbrust, temp[n], top = n, prority[n];
    int i;

    for (i = 0; i < n; i++)
    {
        p[i] = i;
        stack[i] = i;
        cout << "\nPlease enter the Arrival Time: ";
        cin >> arrival[i];
        cout << "Please enter the CPU Brust Time: ";
        cin >> brust[i];
        cout << "Please enter the Priority Time: ";
        cin >> prority[i];
        temp[i] = arrival[i];
        sbrust = brust[i] + sbrust;
    }

    for (i = 0; i < sbrust; i++)
    {
        //section 1
```

```
        proc = stack[0];
        if (temp[proc] == i)
            twaiting = 0;
        else
            twaiting = i - (temp[proc]);
        temp[proc] = i + 1;
        wait[proc] = wait[proc] + twaiting;
        waiting = waiting + (twaiting);
        brust[proc] = brust[proc] - 1;

        if (brust[proc] == 0)
        {
            for (int x = 0; x < top - 1; x++)
                stack[x] = stack[x + 1];
            top = top - 1;
        }

        for (int z = 0; z < top - 1; z++)
        {
            if ((prority[stack[0]] > prority[stack[z + 1]]) && (arrival[stack[z
+ 1]] <= i + 1))
            {
                int t = stack[0];
                stack[0] = stack[z + 1];
                stack[z + 1] = t;
            }
        }
    }

    cout << "\nAverage Waiting Time : " << waiting / n;
    float tu = (sbrust + waiting) / n;
    cout << endl
         << "Average Turnaround Time : " << tu << endl;
    return 0;
}
```

OUTPUT

```
PS C:\Users\nisha\Desktop\OSPracticals> g++ .\Practical10.cpp -o .\Practical10
PS C:\Users\nisha\Desktop\OSPracticals> .\Practical10
Please enter the number of processes: 3

Please enter the Arrival Time: 0
Please enter the CPU Brust Time: 2
Please enter the Priority Time: 1

Please enter the Arrival Time: 0
Please enter the CPU Brust Time: 3
Please enter the Priority Time: 4

Please enter the Arrival Time: 0
Please enter the CPU Brust Time: 1
Please enter the Priority Time: 2

Average Waiting Time : 1.66667
Average Turnaround Time : 3.66667
PS C:\Users\nisha\Desktop\OSPracticals>
```