**Week - 3**

**(22 June 2023)**

**Experiment - 3**

**Question:**

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time:

**(a)** Priority (Non-pre-emptive)

**(b)** Round Robin (Non-pre-emptive)

**Program:**

1. **Priority (Non-pre-emptive)**

#include<stdio.h>

#include<stdlib.h>

struct process {

int process\_id;

int burst\_time;

int priority;

int waiting\_time;

int turnaround\_time;

};

void find\_average\_time(struct process[], int);

void priority\_scheduling(struct process[], int);

int main()

{

int n, i;

struct process proc[10];

printf("Enter the number of processes: ");

scanf("%d", &n);

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

{

printf("\nEnter the process ID: ");

scanf("%d", &proc[i].process\_id);

printf("Enter the burst time: ");

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

printf("Enter the priority: ");

scanf("%d", &proc[i].priority);

}

priority\_scheduling(proc, n);

return 0;

}

void find\_waiting\_time(struct process proc[], int n, int wt[])

{

int i;

wt[0] = 0;

for(i = 1; i< n; i++)

{

wt[i] = proc[i - 1].burst\_time + wt[i - 1];

}

}

void find\_turnaround\_time(struct process proc[], int n, int wt[], int tat[])

{

int i;

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

{

tat[i] = proc[i].burst\_time + wt[i];

}

}

void find\_average\_time(struct process proc[], int n)

{

int wt[10], tat[10], total\_wt = 0, total\_tat = 0, i;

find\_waiting\_time(proc, n, wt);

find\_turnaround\_time(proc, n, wt, tat);

printf("\nProcess ID\tBurst Time\tPriority\tWaiting Time\tTurnaround Time");

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", proc[i].process\_id, proc[i].burst\_time, proc[i].priority, wt[i], tat[i]);

}

printf("\n\nAverage Waiting Time = %f", (float)total\_wt/n);

printf("\nAverage Turnaround Time = %f\n", (float)total\_tat/n);

}

void priority\_scheduling(struct process proc[], int n)

{

int i, j, pos;

struct process temp;

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

{

pos = i;

for(j = i + 1; j < n; j++)

{

if(proc[j].priority< proc[pos].priority)

pos = j;

}

temp = proc[i];

proc[i] = proc[pos];

proc[pos] = temp;

}

find\_average\_time(proc, n);

}

**(b) Round Robin (Non-pre-emptive)**

#include <stdio.h>

#include <stdbool.h>

int turnarroundtime(int processes[], int n, int bt[], int wt[], int tat[]) {

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

tat[i] = bt[i] + wt[i];

return 1;

}

int waitingtime(int processes[], int n, int bt[], int wt[], int quantum)

{

int rem\_bt[n];

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

rem\_bt[i] = bt[i];

int t = 0;

while (1)

{

bool done = true;

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

{

if (rem\_bt[i] > 0)

{

done = false;

if (rem\_bt[i] > quantum)

{

t += quantum;

rem\_bt[i] -= quantum;

}

else

{

t = t + rem\_bt[i];

wt[i] = t - bt[i];

rem\_bt[i] = 0;

}

}

}

if (done == true)

break;

}

return 1;

}

int findavgTime(int processes[], int n, int bt[], int quantum) {

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

waitingtime(processes, n, bt, wt, quantum);

turnarroundtime(processes, n, bt, wt, tat);

printf("\n\nProcesses\t\t Burst Time\t\t Waiting Time\t\t turnaround time\n");

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

printf("\n\t%d\t\t\t%d\t\t\t%d\t\t\t%d\n",i+1, bt[i], wt[i], tat[i]);

}

printf("\nAverage waiting time = %f", (float)total\_wt / (float)n);

printf("\nAverage turnaround time = %f", (float)total\_tat / (float)n);

return 1;

}

int main()

{

int n, processes[n], burst\_time[n], quantum;

printf("Enter the Number of Processes: ");

scanf("%d",&n);

printf("\nEnter the quantum time: ");

scanf("%d",&quantum);

int i=0;

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

{

printf("\nEnter the process: ");

scanf("%d",&processes[i]);

printf("Enter the Burst Time:");

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

}

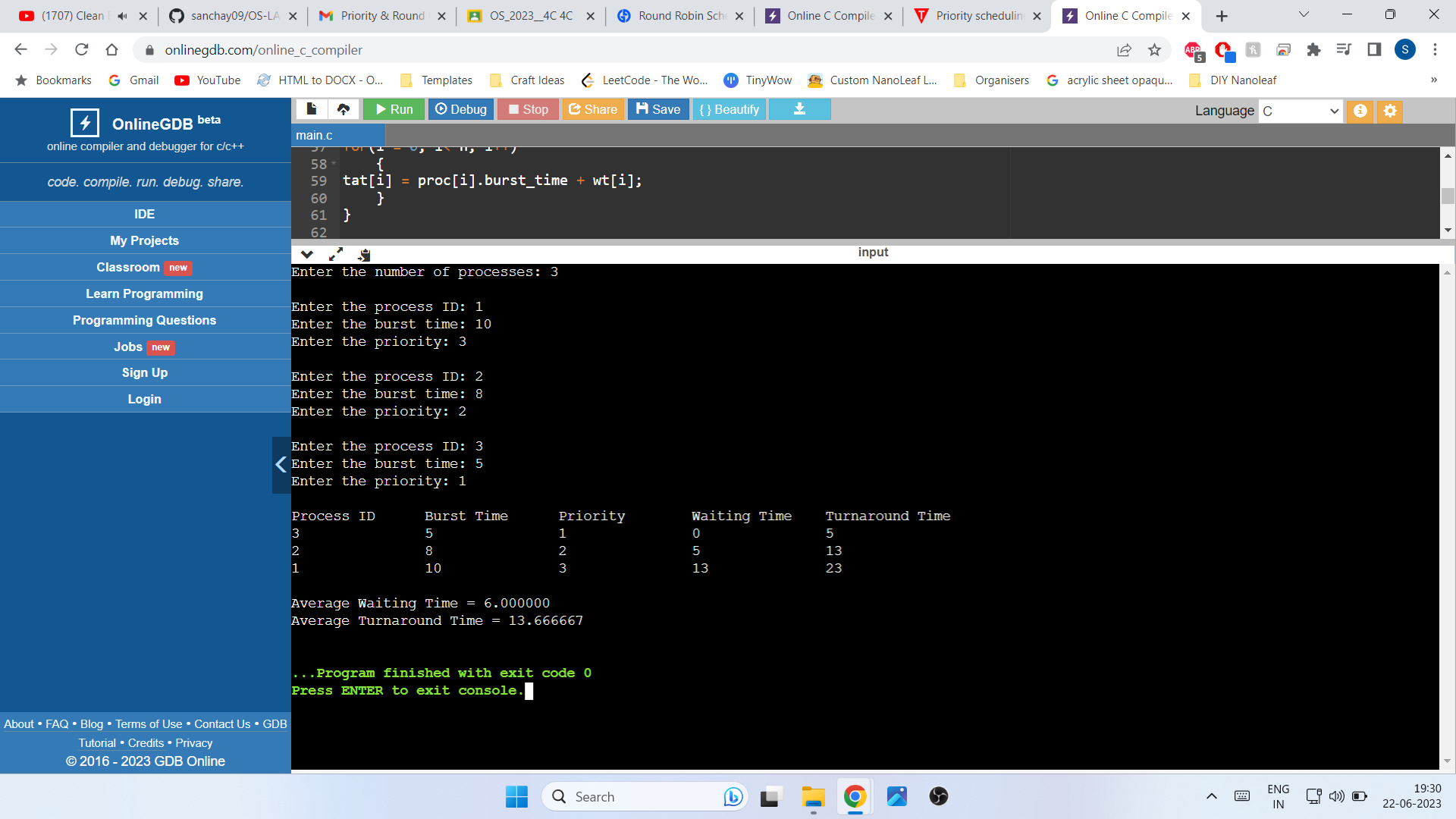
findavgTime(processes, n, burst\_time, quantum);

return 0;

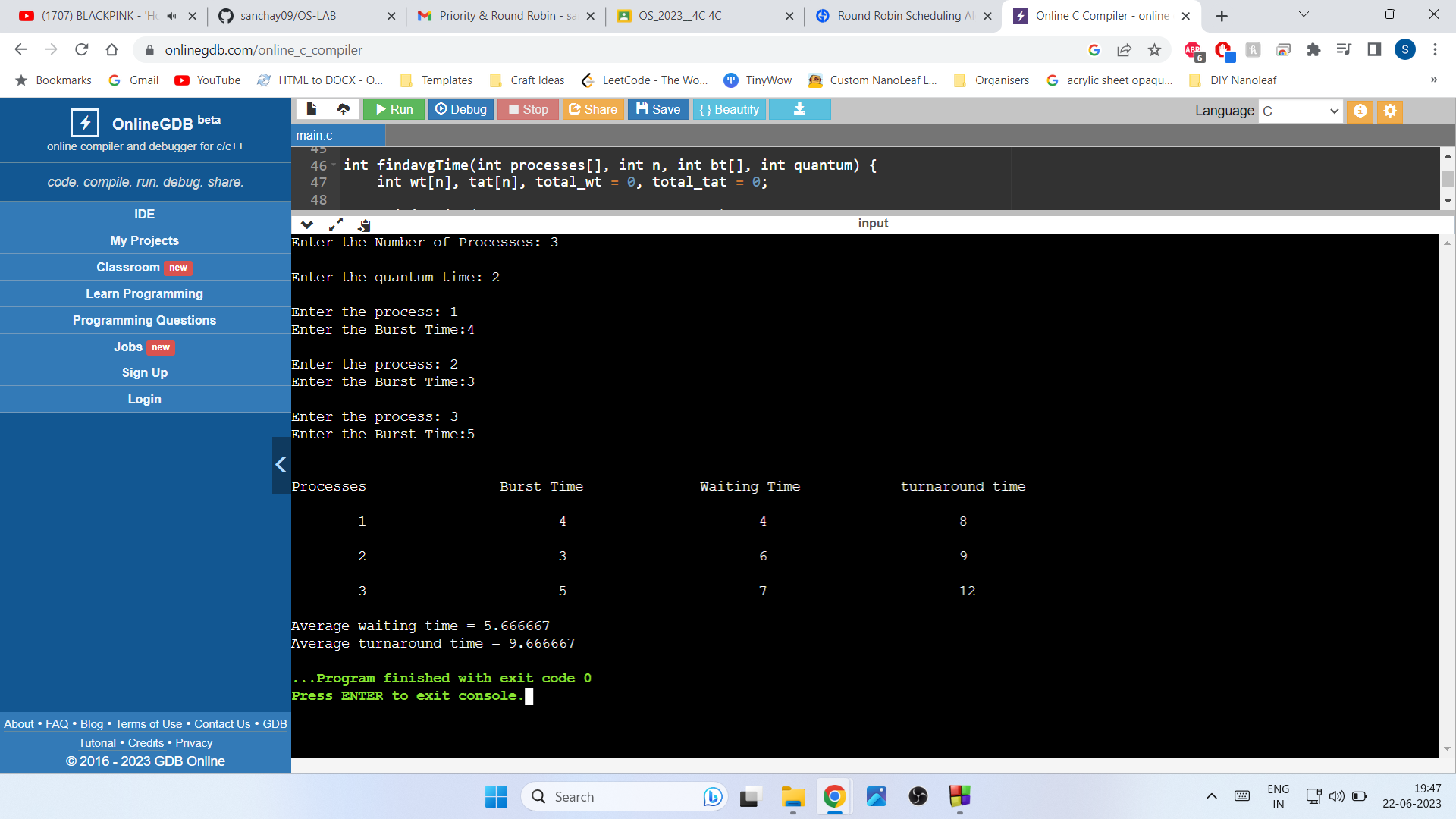
}

**Output:**

1. **Priority (Non-pre-emptive)**



1. **Round Robin (Non-pre-emptive)**



**Observation Book Pictures:**

