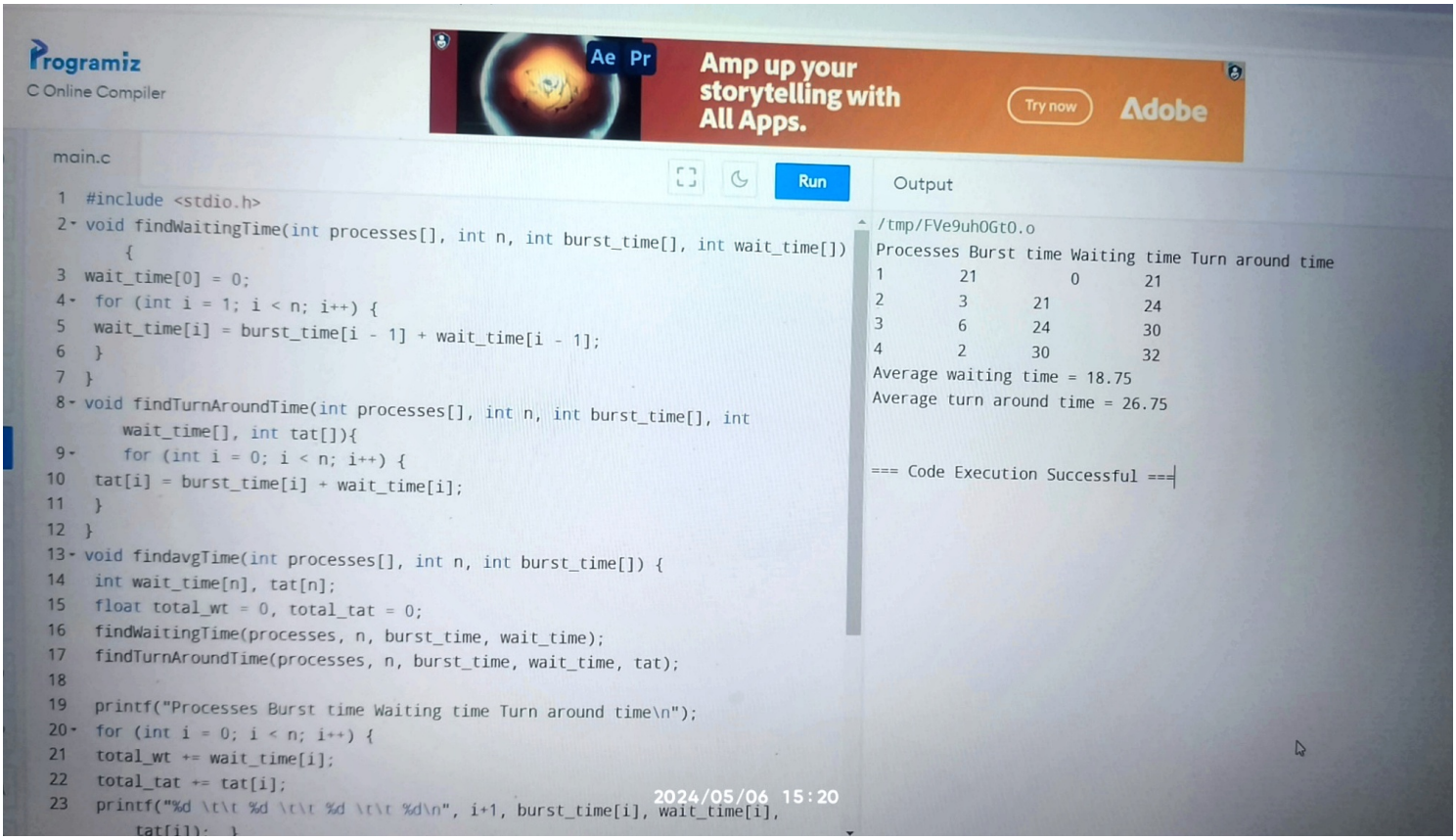
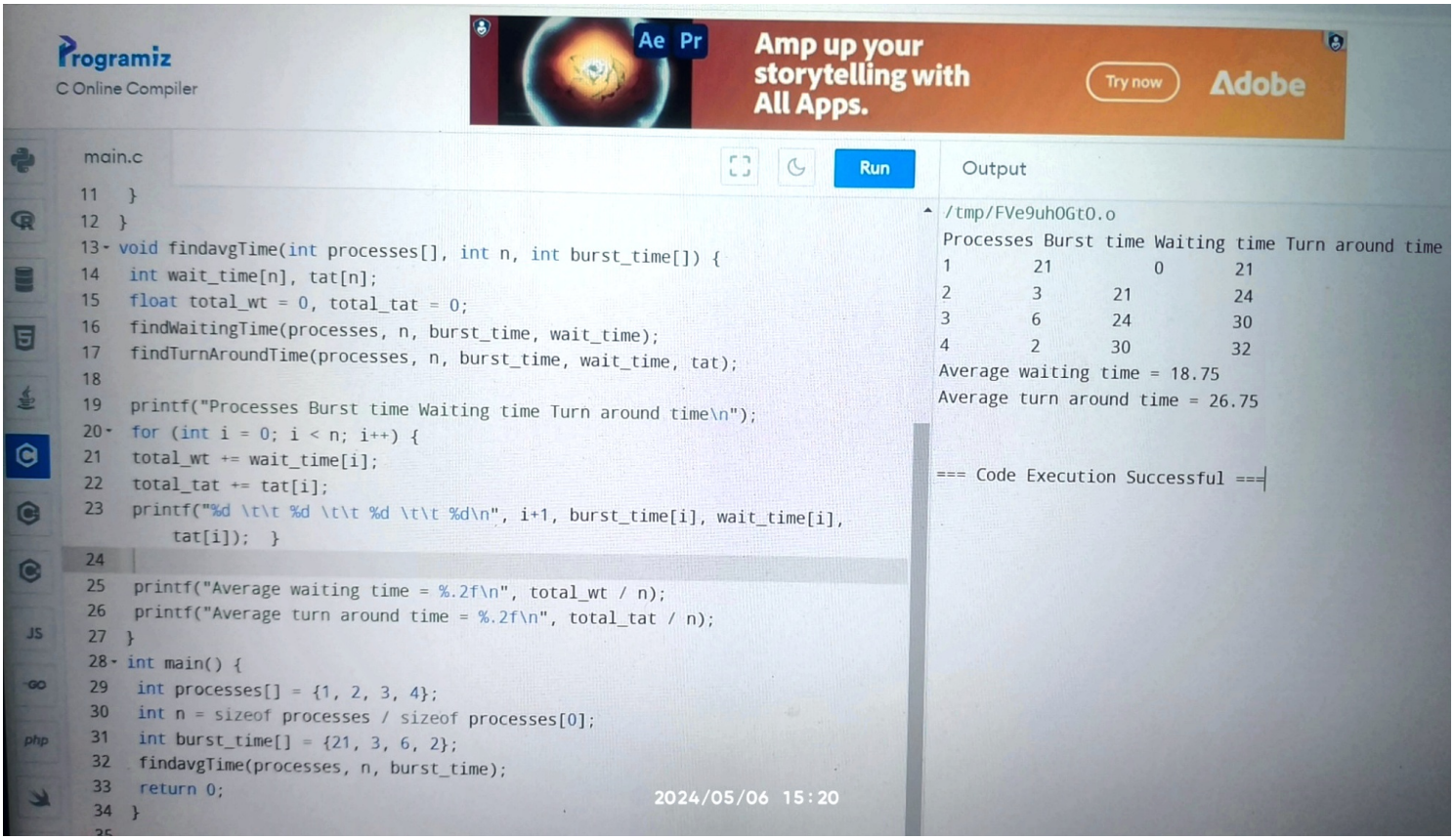
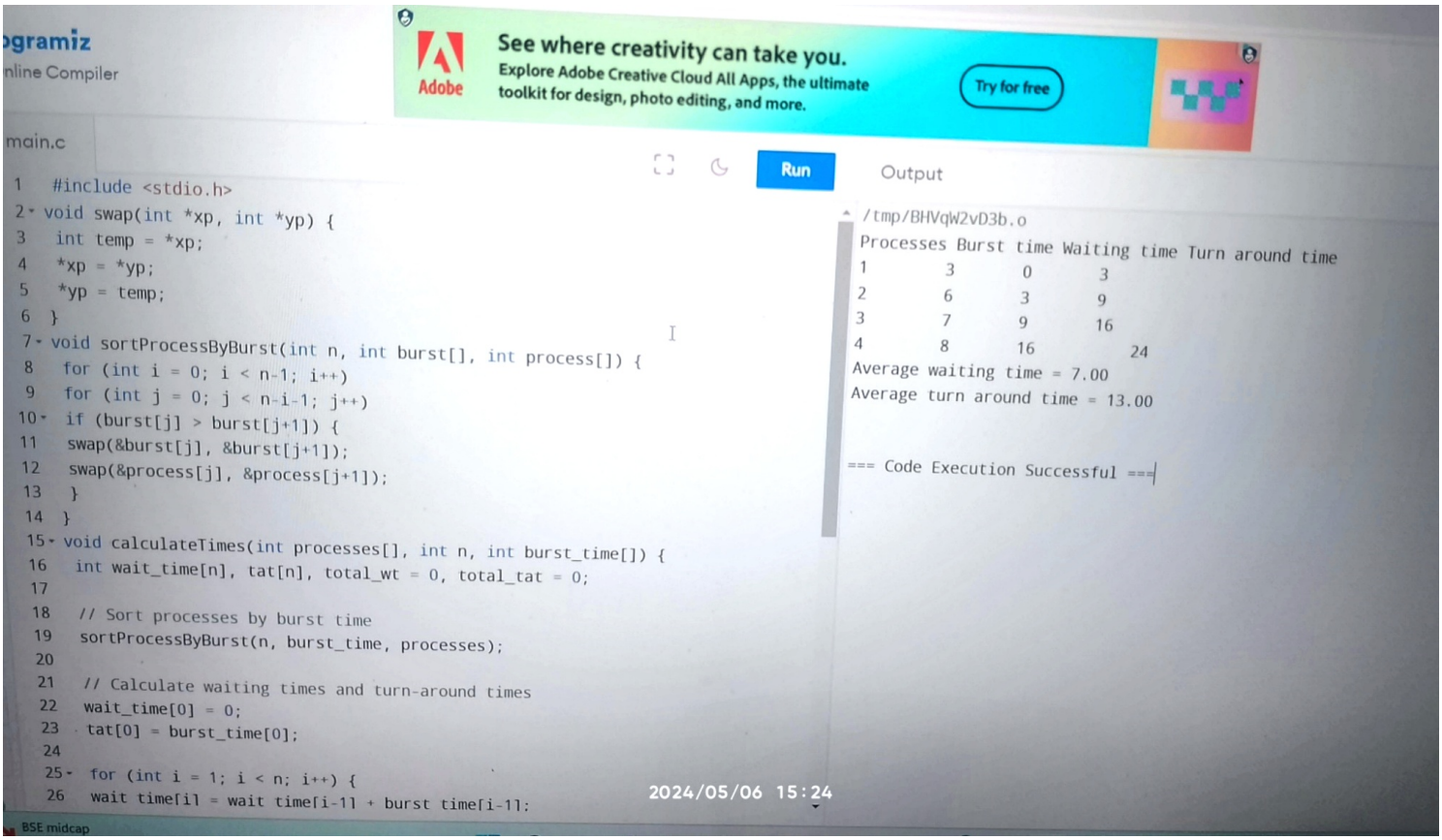
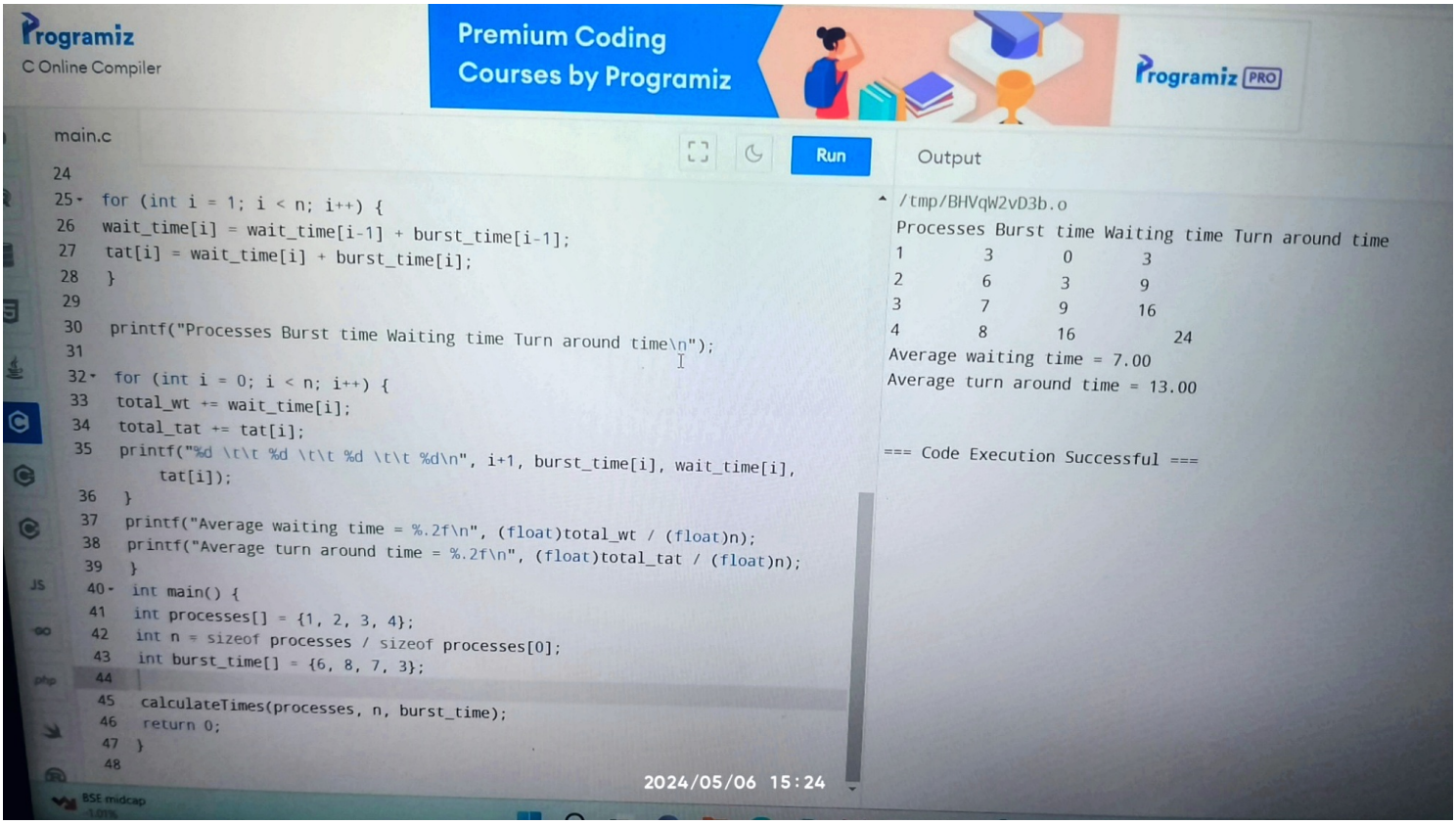
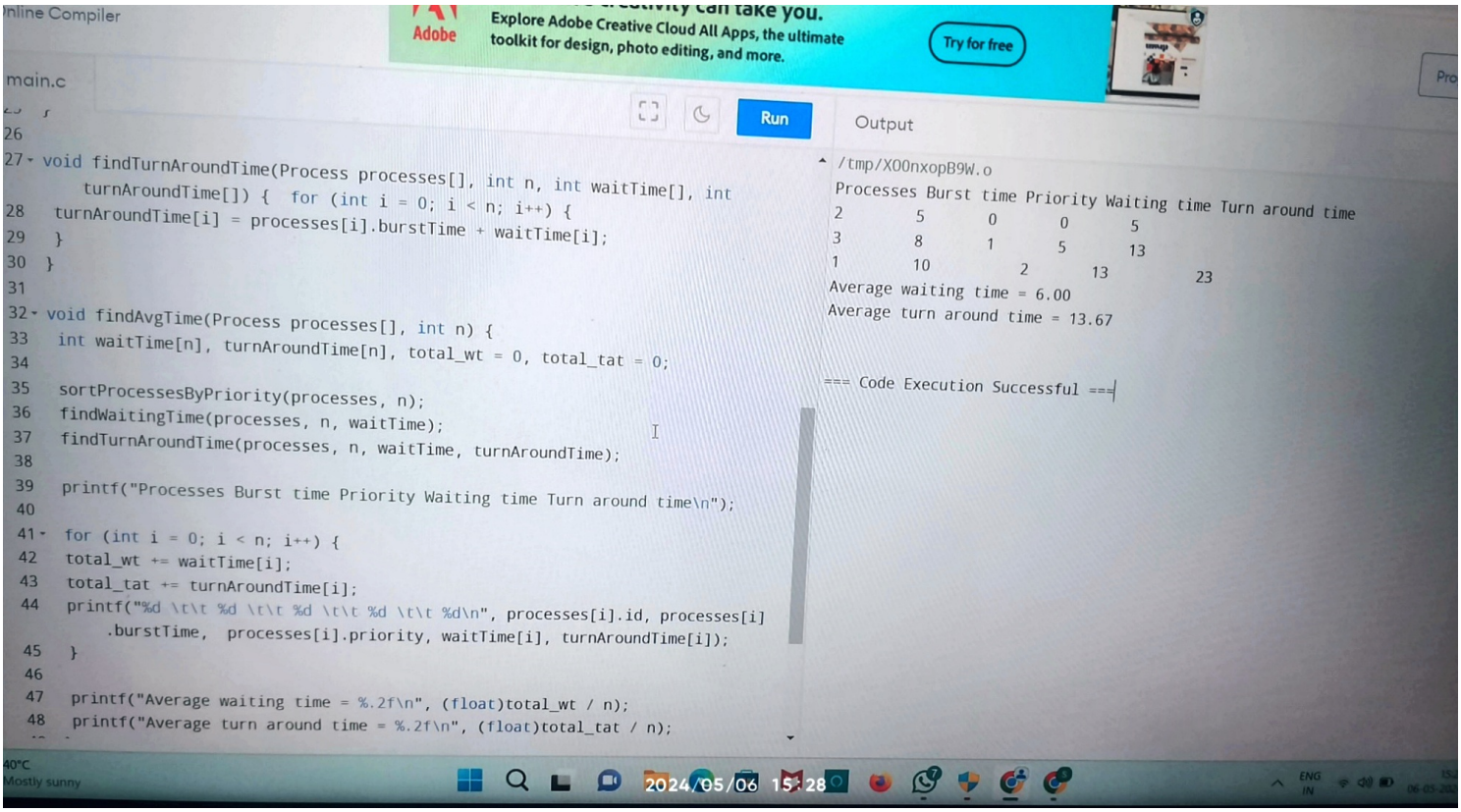
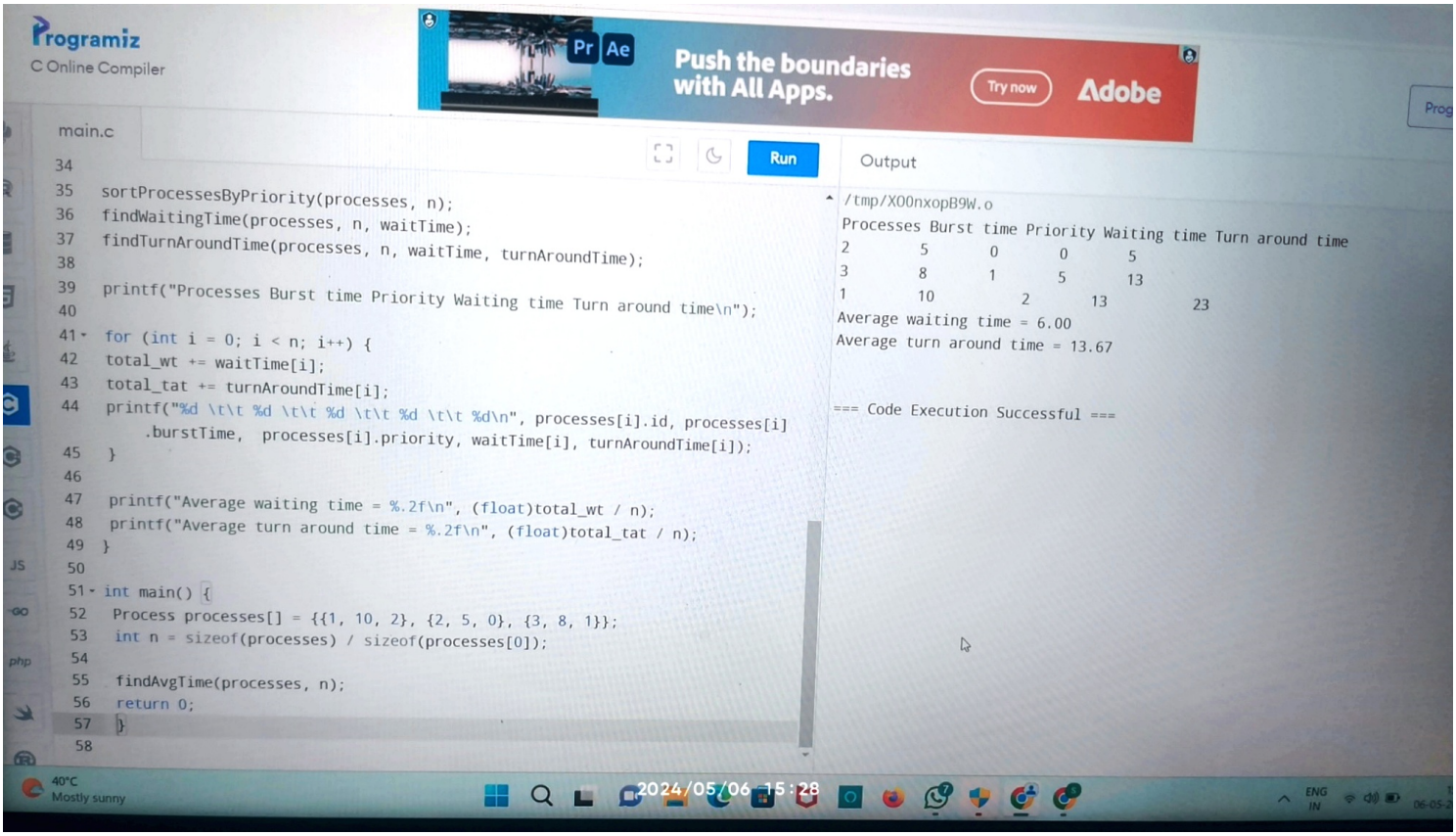

Programiz
C Online Compiler
Amp up your storytelling with All Apps.
Try now
Adobe
main.c
Run
Output
#include <stdio.h>
void findWaitingTime(int processes[], int n, int burst_time[], int wait_time[])
{
wait_time[0] = 0;
for (int i = 1; i < n; i++) {
wait_time[i] = burst_time[i - 1] + wait_time[i - 1];
}
}
void findTurnAroundTime(int processes[], int n, int burst_time[], int
wait_time[], int tat[]){
for (int i = 0; i < n; i++) {
tat[i] = burst_time[i] + wait_time[i];
}
}
void findavgTime(int processes[], int n, int burst_time[]) {
int wait_time[n], tat[n];
float total_wt = 0, total_tat = 0;
findWaitingTime(processes, n, burst_time, wait_time);
findTurnAroundTime(processes, n, burst_time, wait_time, tat);
printf("Processes Burst time Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += wait_time[i];
total_tat += tat[i];
printf("%d \t\t %d \t\t %d \t\t %d\n", i+1, burst_time[i], wait_time[i],
/tmp/FVe9uhOGtO.o
Processes Burst time Waiting time Turn around time
1 21 0 21
2 3 21 24
3 6 24 30
4 2 30 32
Average waiting time = 18.75
Average turn around time = 26.75
=== Code Execution Successful ===
2024/05/06 15:20

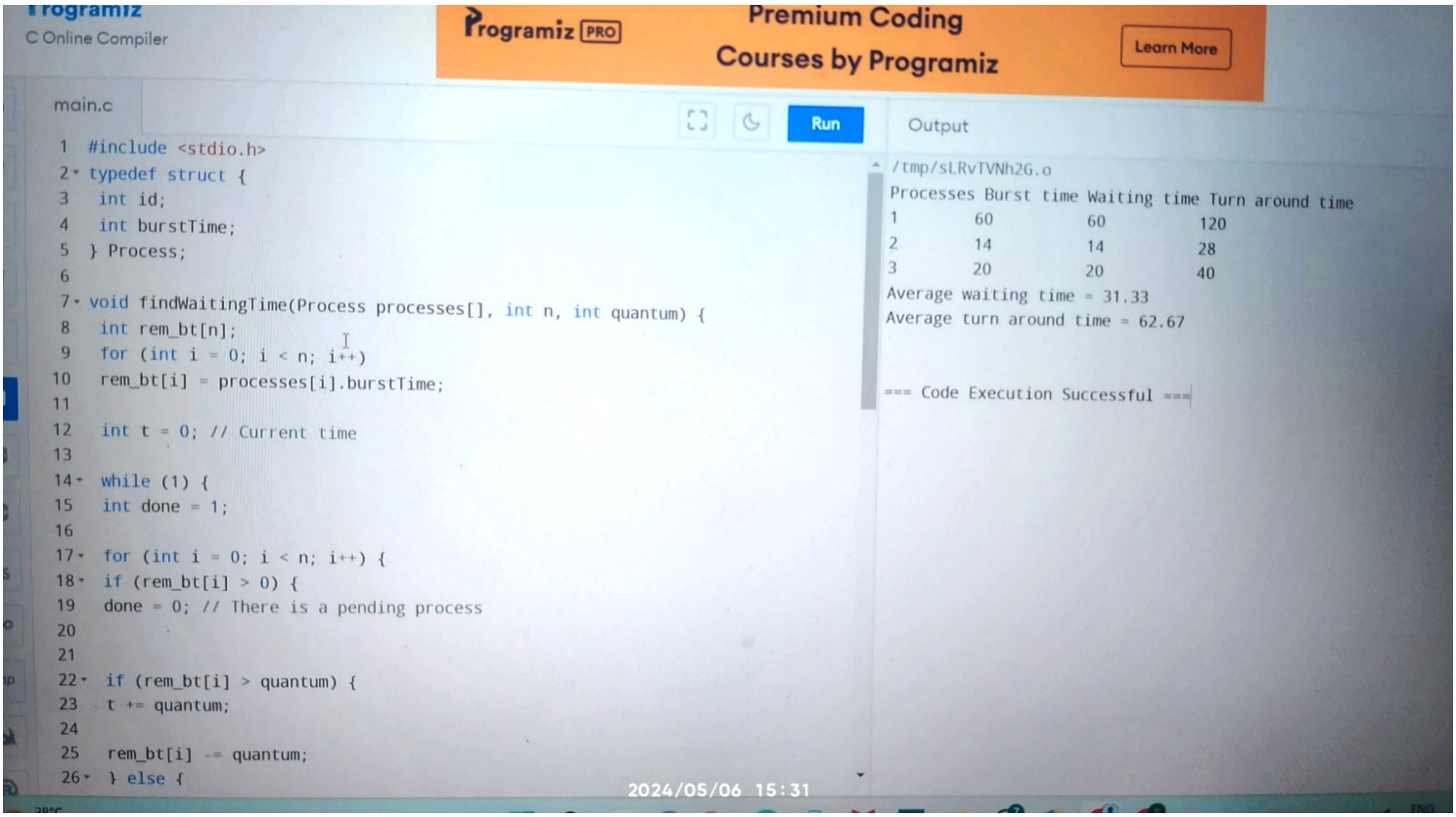

Programiz
C Online Compiler
Amp up your storytelling with All Apps.
Try now
Adobe
main.c
Run
Output
}
}
void findavgTime(int processes[], int n, int burst_time[]) {
int wait_time[n], tat[n];
float total_wt = 0, total_tat = 0;
findWaitingTime(processes, n, burst_time, wait_time);
findTurnAroundTime(processes, n, burst_time, wait_time, tat);
printf("Processes Burst time Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += wait_time[i];
total_tat += tat[i];
printf("%d \t\t %d \t\t %d \t\t %d\n", i+1, burst_time[i], wait_time[i],
tat[i]); }
printf("Average waiting time = %.2f\n", total_wt / n);
printf("Average turn around time = %.2f\n", total_tat / n);
}
int main() {
int processes[] = {1, 2, 3, 4};
int n = sizeof processes / sizeof processes[0];
int burst_time[] = {21, 3, 6, 2};
findavgTime(processes, n, burst_time);
return 0;
}
2024/05/06 15:20
/tmp/FVe9uhOGtO.o
Processes Burst time Waiting time Turn around time
1 21 0 21
2 3 21 24
3 6 24 30
4 2 30 32
Average waiting time = 18.75
Average turn around time = 26.75
=== Code Execution Successful ===

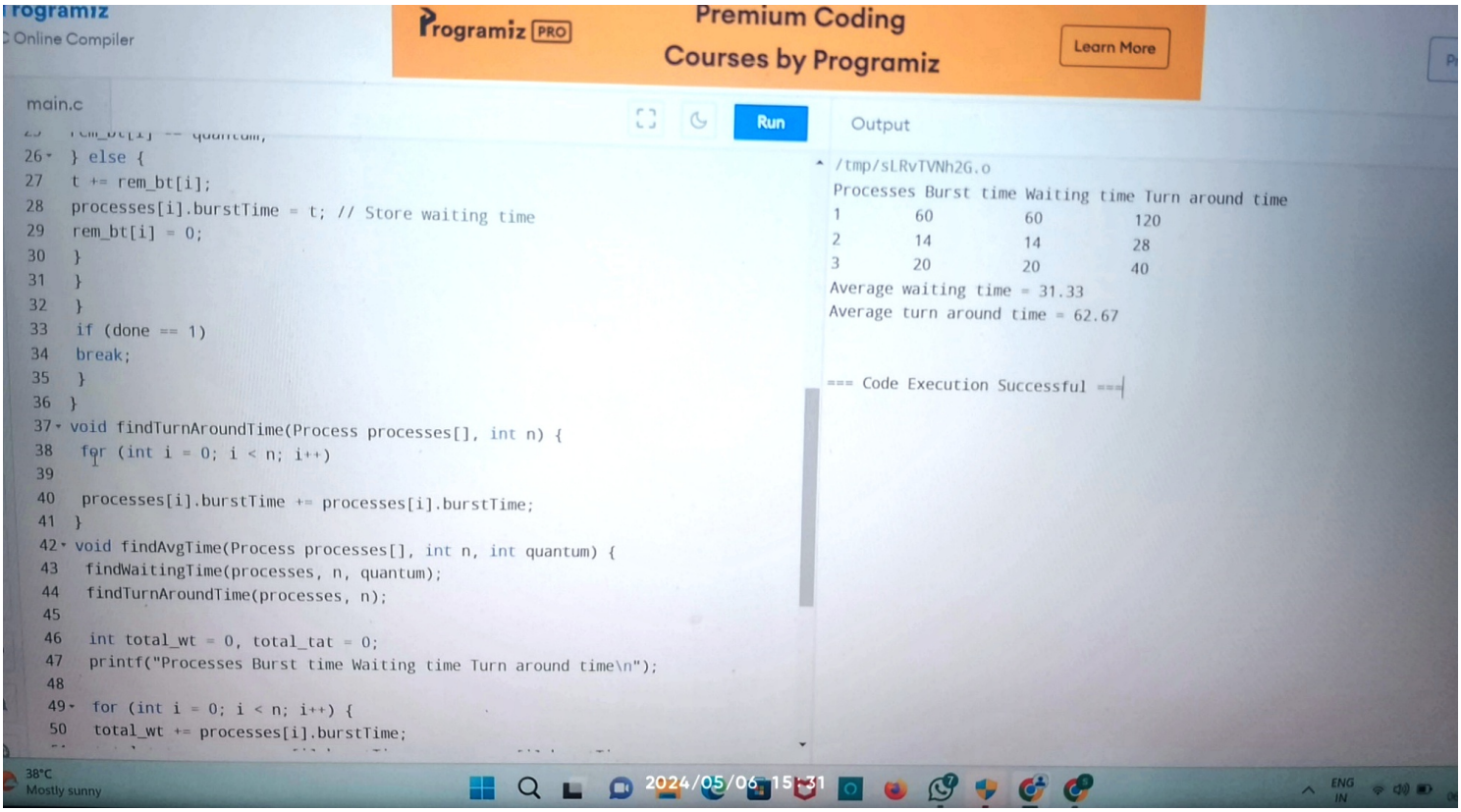

ogramiz
nline Compiler
See where creativity can take you.
Explore Adobe Creative Cloud All Apps, the ultimate toolkit for design, photo editing, and more.
Try for free
Adobe
main.c
Run
Output
#include <stdio.h>
void swap(int *xp, int *yp) {
int temp = *xp;
*xp = *yp;
*yp = temp;
}
void sortProcessByBurst(int n, int burst[], int process[]) {
for (int i = 0; i < n-1; i++)
for (int j = 0; j < n-i-1; j++)
if (burst[j] > burst[j+1]) {
swap(&burst[j], &burst[j+1]);
swap(&process[j], &process[j+1]);
}
}
void calculateTimes(int processes[], int n, int burst_time[]) {
int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
// Sort processes by burst time
sortProcessByBurst(n, burst_time, processes);
// Calculate waiting times and turn-around times
wait_time[0] = 0;
tat[0] = burst_time[0];
for (int i = 1; i < n; i++) {
wait time[i] = wait time[i-1] + burst time[i-1];
/tmp/BHVqW2vD3b.o
Processes Burst time Waiting time Turn around time
1 3 0 3
2 6 3 9
3 7 9 16
4 8 16 24
Average waiting time = 7.00
Average turn around time = 13.00
=== Code Execution Successful ===
2024/05/06 15:24
BSE midcap

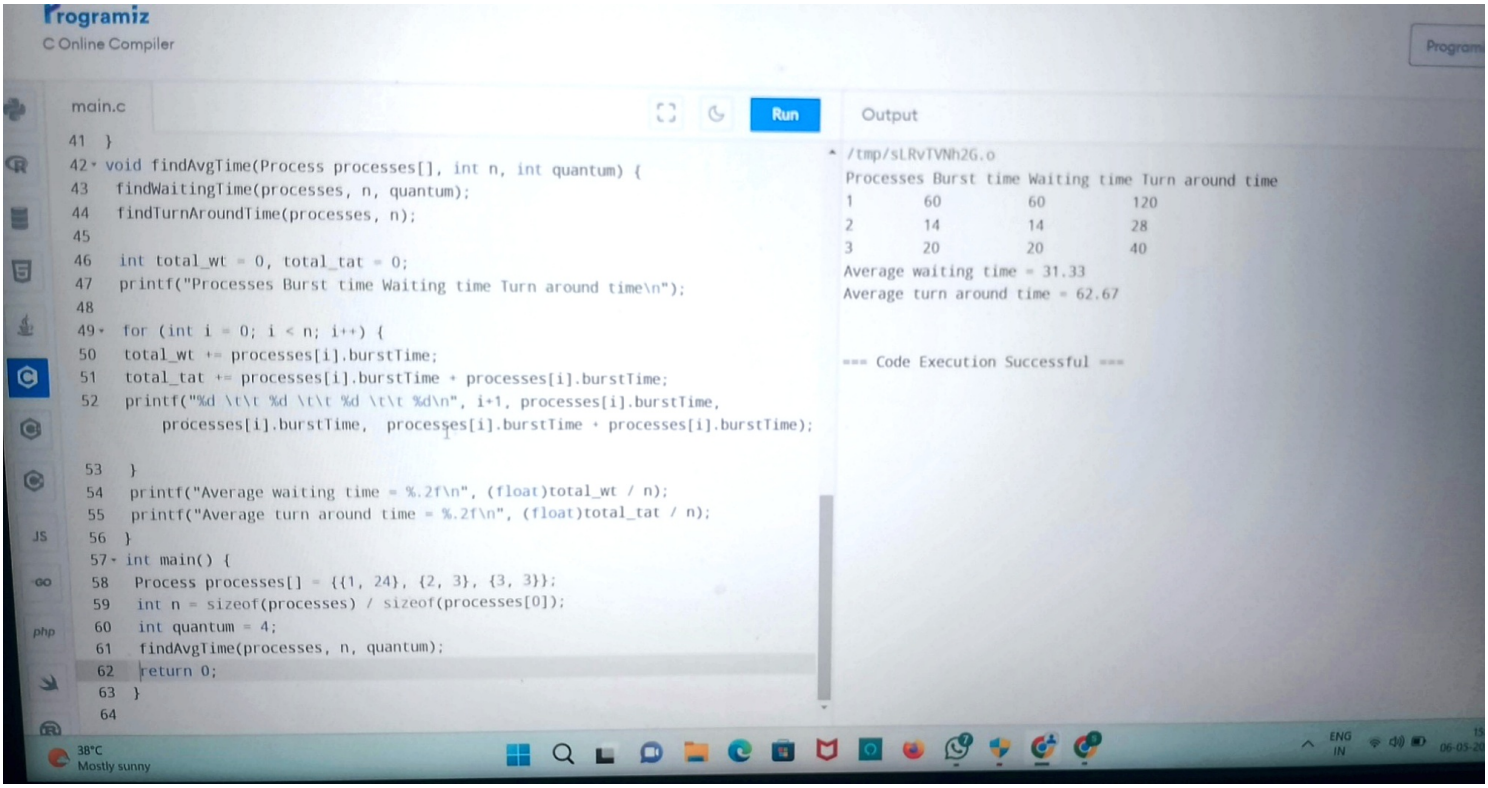

Programiz
C Online Compiler
Premium Coding
Courses by Programiz
Programiz PRO
main.c
Run
Output
for (int i = 1; i < n; i++) {
wait_time[i] = wait_time[i-1] + burst_time[i-1];
tat[i] = wait_time[i] + burst_time[i];
}
printf("Processes Burst time Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += wait_time[i];
total_tat += tat[i];
printf("%d \t\t %d \t\t %d \t\t %d\n", i+1, burst_time[i], wait_time[i],
tat[i]);
}
printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
printf("Average turn around time = %.2f\n", (float)total_tat / (float)n);
}
int main() {
int processes[] = {1, 2, 3, 4};
int n = sizeof processes / sizeof processes[0];
int burst_time[] = {6, 8, 7, 3};
calculateTimes(processes, n, burst_time);
return 0;
}
/tmp/BHVqW2vD3b.o
Processes Burst time Waiting time Turn around time
1 3 0 3
2 6 3 9
3 7 9 16
4 8 16 24
Average waiting time = 7.00
Average turn around time = 13.00
=== Code Execution Successful ===
2024/05/06 15:24

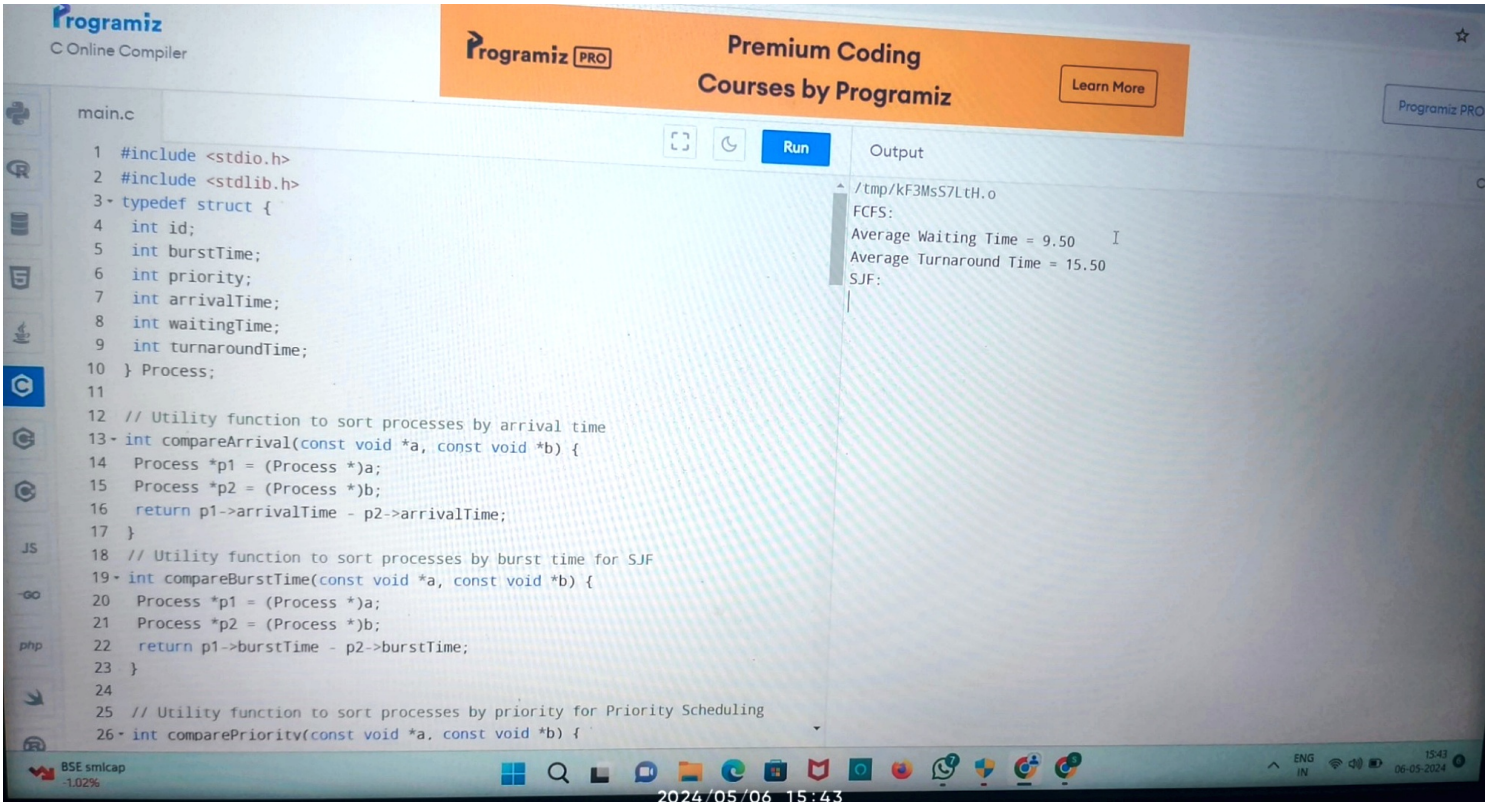

Inline Compiler
Adobe
Explore Adobe Creative Cloud All Apps, the ultimate toolkit for design, photo editing, and more.
Try for free
main.c
Run
Output
void findTurnAroundTime(Process processes[], int n, int waitTime[], int turnAroundTime[]) { for (int i = 0; i < n; i++) {
turnAroundTime[i] = processes[i].burstTime + waitTime[i];
}
}
void findAvgTime(Process processes[], int n) {
int waitTime[n], turnAroundTime[n], total_wt = 0, total_tat = 0;
sortProcessesByPriority(processes, n);
findWaitingTime(processes, n, waitTime);
findTurnAroundTime(processes, n, waitTime, turnAroundTime);
printf("Processes Burst time Priority Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += waitTime[i];
total_tat += turnAroundTime[i];
printf("%d \t\t %d \t\t %d \t\t %d \t\t %d\n", processes[i].id, processes[i].burstTime, processes[i].priority, waitTime[i], turnAroundTime[i]);
}
printf("Average waiting time = %.2f\n", (float)total_wt / n);
printf("Average turn around time = %.2f\n", (float)total_tat / n);
/tmp/XO0nxopB9W.o
Processes Burst time Priority Waiting time Turn around time
2 5 0 0 5
3 8 1 5 13
1 10 2 13 23
Average waiting time = 6.00
Average turn around time = 13.67
=== Code Execution Successful ===
40°C
Mostly sunny
2024/05/06 15:28
ENG
IN

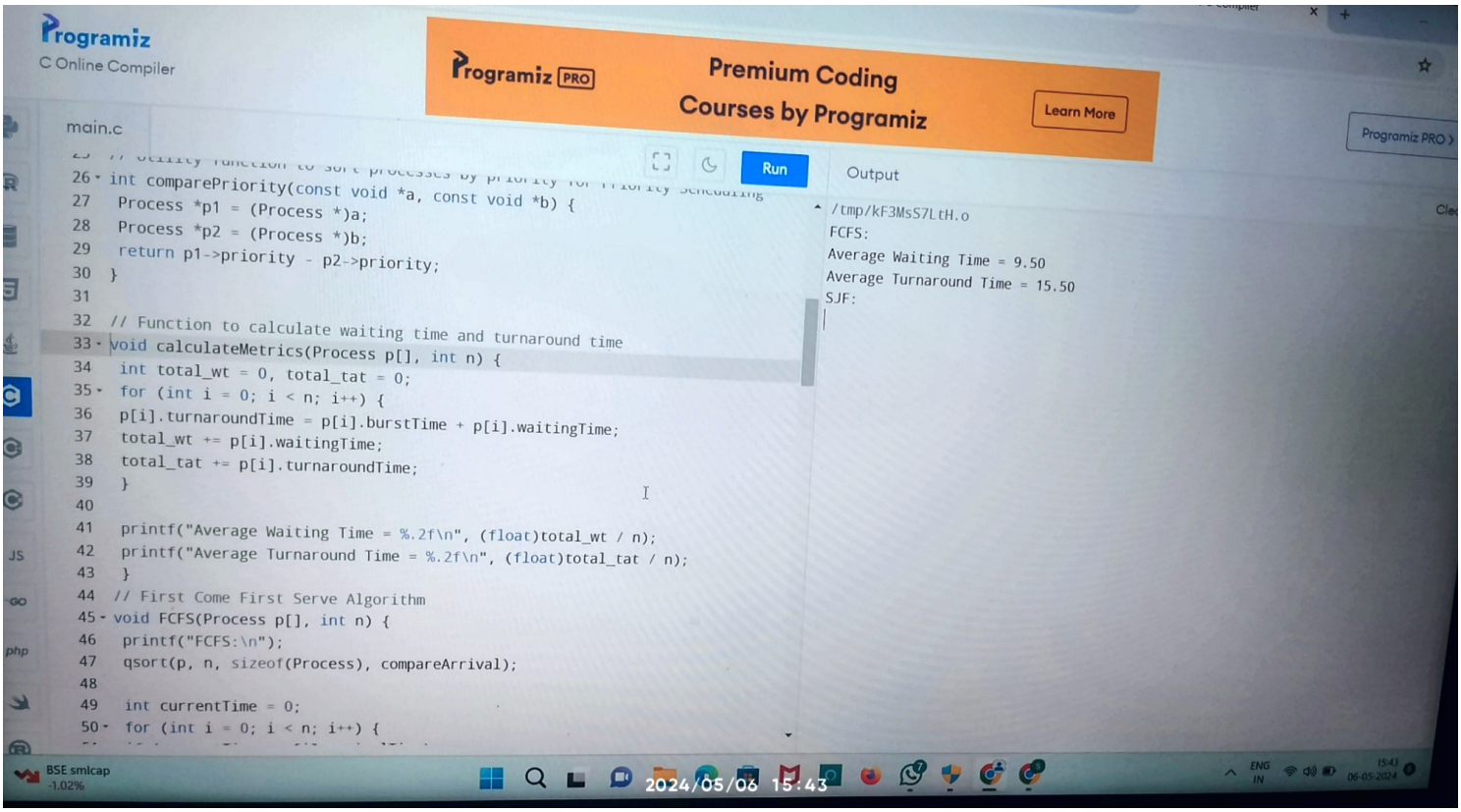

Programiz
C Online Compiler
Push the boundaries with All Apps.
Try now
Adobe
main.c
Run
Output
sortProcessesByPriority(processes, n);
findWaitingTime(processes, n, waitTime);
findTurnAroundTime(processes, n, waitTime, turnAroundTime);
printf("Processes Burst time Priority Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += waitTime[i];
total_tat += turnAroundTime[i];
printf("%d \t\t %d \t\t %d \t\t %d \t\t %d\n", processes[i].id, processes[i]
.burstTime, processes[i].priority, waitTime[i], turnAroundTime[i]);
}
printf("Average waiting time = %.2f\n", (float)total_wt / n);
printf("Average turn around time = %.2f\n", (float)total_tat / n);
}
int main() {
Process processes[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}};
int n = sizeof(processes) / sizeof(processes[0]);
findAvgTime(processes, n);
return 0;
}
/tmp/XO0nxopB9W.o
Processes Burst time Priority Waiting time Turn around time
2 5 0 0 5
3 8 1 5 13
1 10 2 13 23
Average waiting time = 6.00
Average turn around time = 13.67
=== Code Execution Successful ===
40°C
Mostly sunny
2024/05/06 15:28
ENG
IN

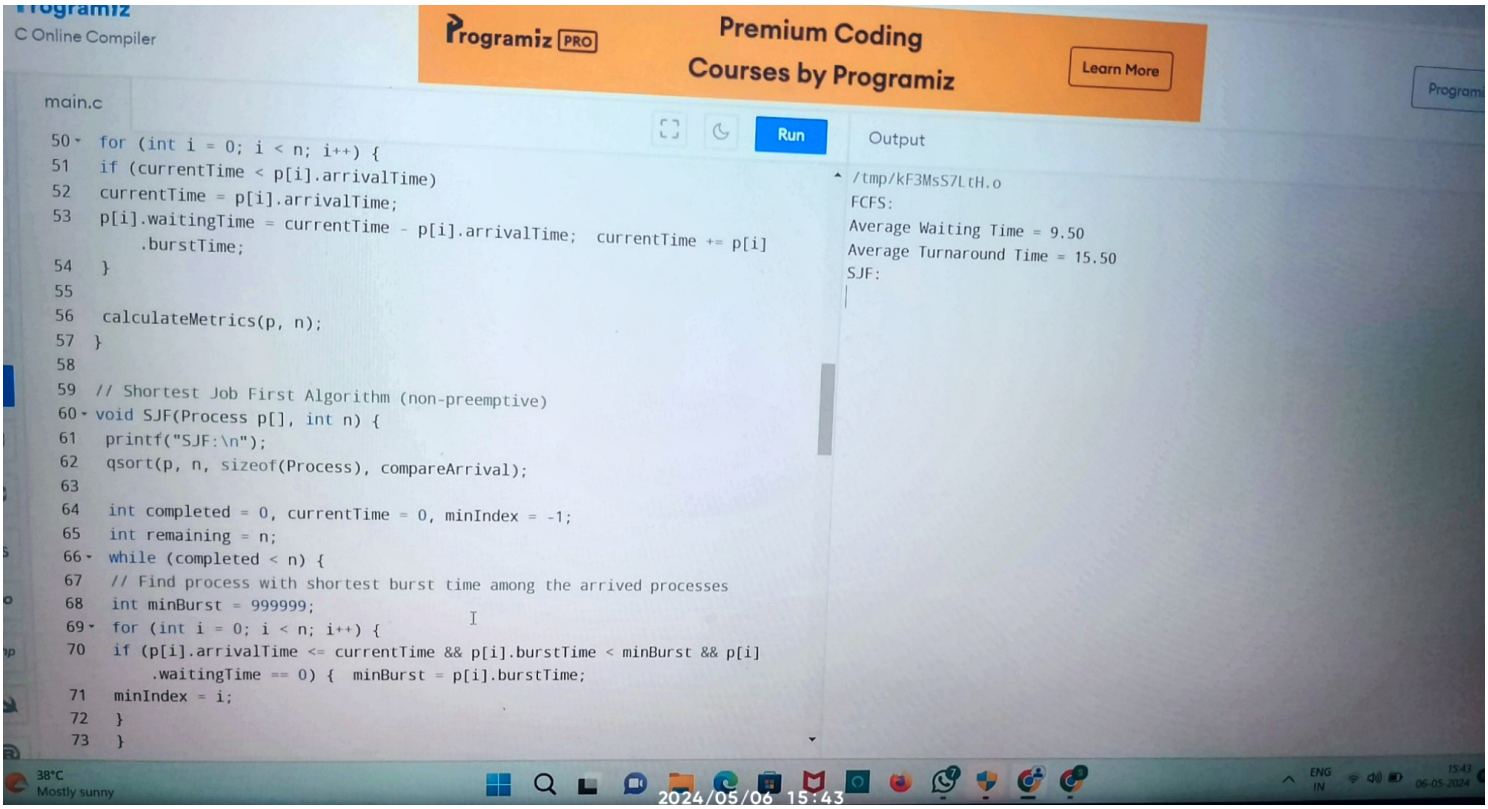

Programiz
C Online Compiler
Programiz PRO
Premium Coding
Courses by Programiz
Learn More
main.c
Run
Output
#include <stdio.h>
typedef struct {
int id;
int burstTime;
} Process;
void findWaitingTime(Process processes[], int n, int quantum) {
int rem_bt[n];
for (int i = 0; i < n; i++)
rem_bt[i] = processes[i].burstTime;
int t = 0; // Current time
while (1) {
int done = 1;
for (int i = 0; i < n; i++) {
if (rem_bt[i] > 0) {
done = 0; // There is a pending process
if (rem_bt[i] > quantum) {
t += quantum;
rem_bt[i] -= quantum;
} else {
/tmp/sLRvTVNh2G.o
Processes Burst time Waiting time Turn around time
1 60 60 120
2 14 14 28
3 20 20 40
Average waiting time = 31.33
Average turn around time = 62.67
=== Code Execution Successful ===
2024/05/06 15:31

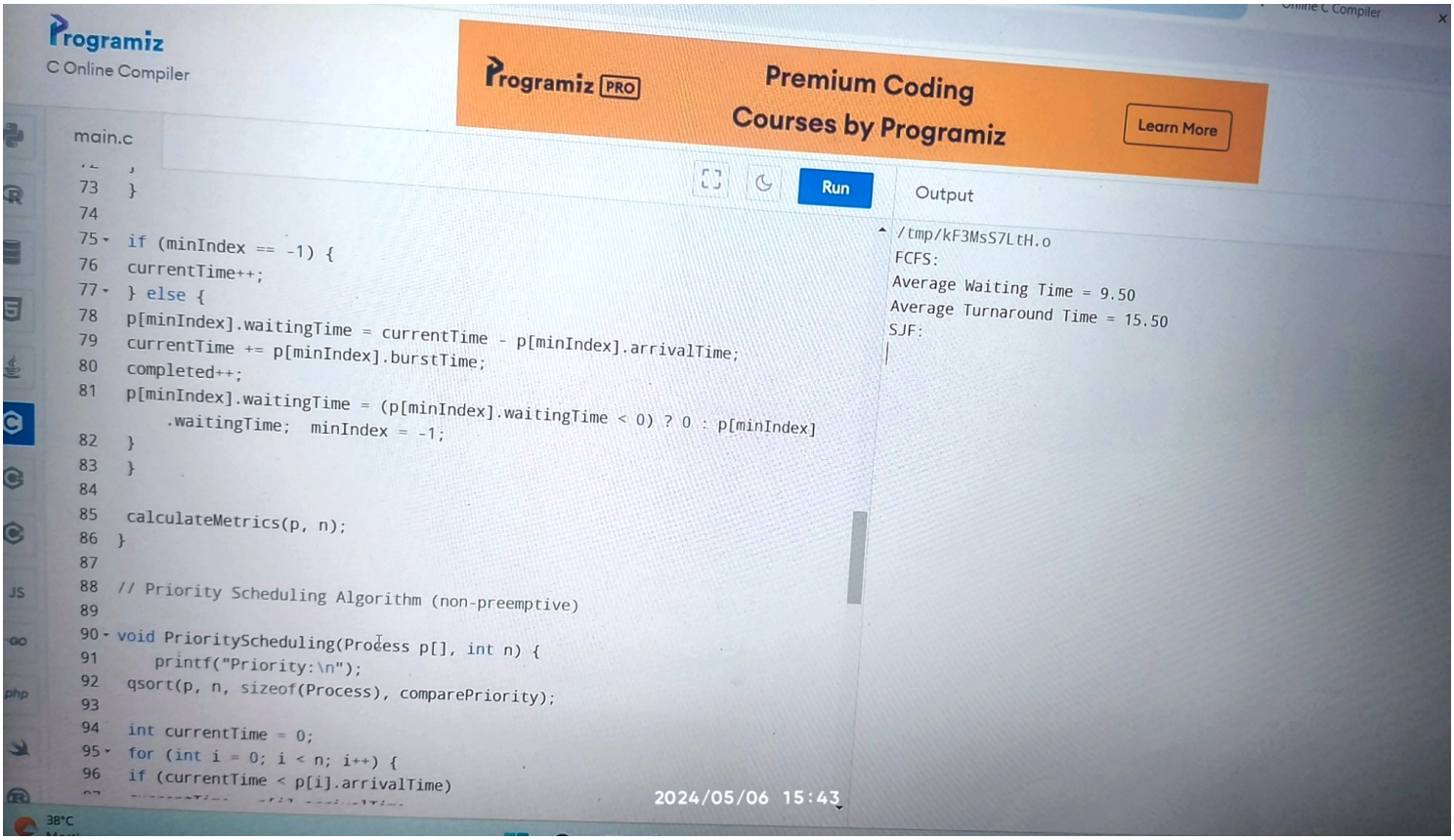

Programiz
Online Compiler
Programiz PRO
Premium Coding
Courses by Programiz
Learn More
main.c
Run
Output
} else {
t += rem_bt[i];
processes[i].burstTime = t; // Store waiting time
rem_bt[i] = 0;
}
}
}
if (done == 1)
break;
}
}
void findTurnAroundTime(Process processes[], int n) {
for (int i = 0; i < n; i++)
processes[i].burstTime += processes[i].burstTime;
}
void findAvgTime(Process processes[], int n, int quantum) {
findWaitingTime(processes, n, quantum);
findTurnAroundTime(processes, n);
int total_wt = 0, total_tat = 0;
printf("Processes Burst time Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += processes[i].burstTime;
/tmp/sLRvTVNh2G.o
Processes Burst time Waiting time Turn around time
1 60 60 120
2 14 14 28
3 20 20 40
Average waiting time = 31.33
Average turn around time = 62.67
=== Code Execution Successful ===
38°C
Mostly sunny
2024/05/06 15:31
ENG
IN

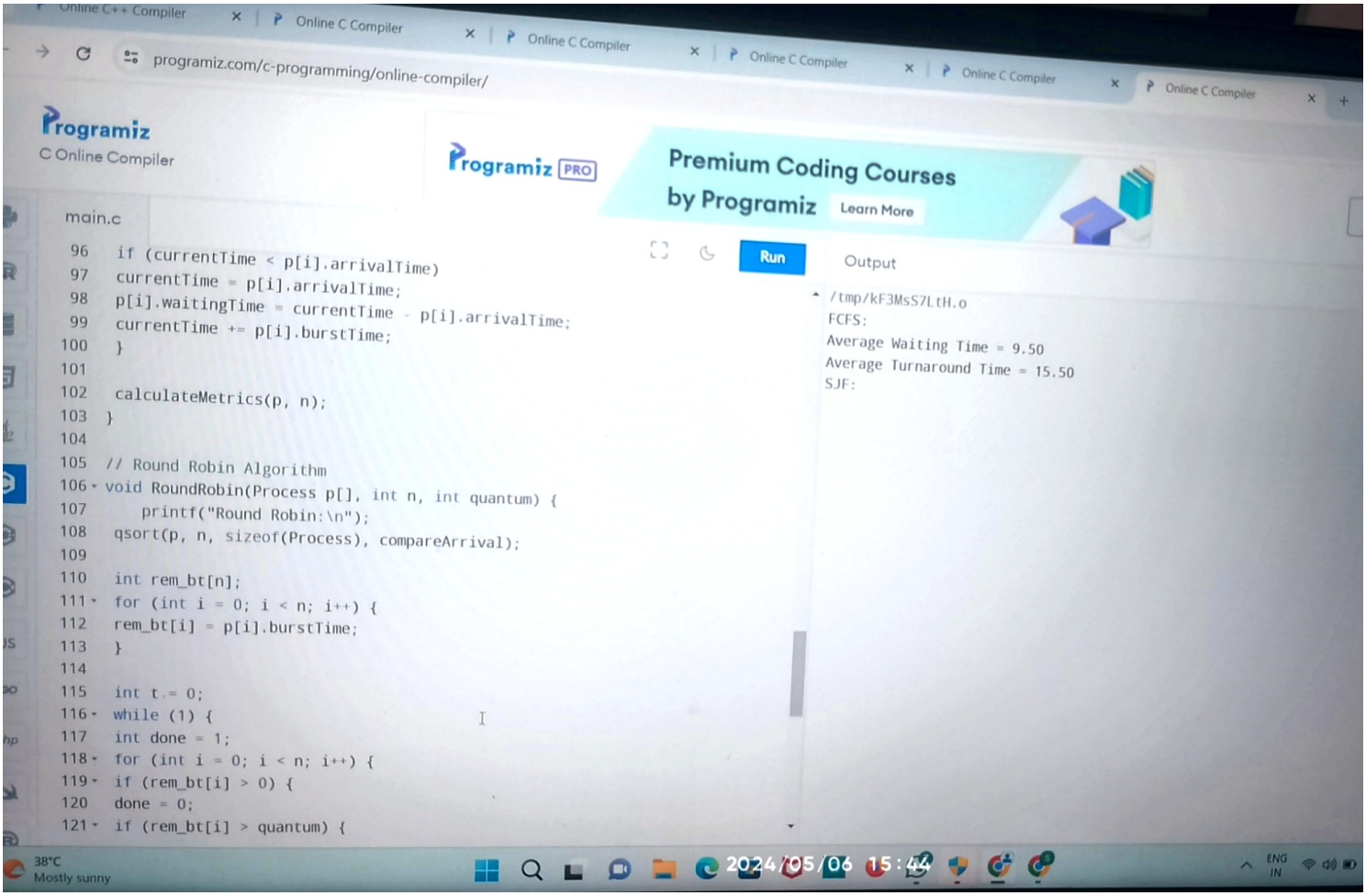

Programiz
C Online Compiler
main.c
Run
Output
}
void findAvgTime(Process processes[], int n, int quantum) {
findWaitingTime(processes, n, quantum);
findTurnAroundTime(processes, n);
int total_wt = 0, total_tat = 0;
printf("Processes Burst time Waiting time Turn around time\n");
for (int i = 0; i < n; i++) {
total_wt += processes[i].burstTime;
total_tat += processes[i].burstTime + processes[i].burstTime;
printf("%d \t\t %d \t\t %d \t\t %d\n", i+1, processes[i].burstTime,
processes[i].burstTime, processes[i].burstTime + processes[i].burstTime);
}
printf("Average waiting time = %.2f\n", (float)total_wt / n);
printf("Average turn around time = %.2f\n", (float)total_tat / n);
}
int main() {
Process processes[] = {{1, 24}, {2, 3}, {3, 3}};
int n = sizeof(processes) / sizeof(processes[0]);
int quantum = 4;
findAvgTime(processes, n, quantum);
return 0;
}
/tmp/sLRvTVNh2G.o
Processes Burst time Waiting time Turn around time
1 60 60 120
2 14 14 28
3 20 20 40
Average waiting time = 31.33
Average turn around time = 62.67
=== Code Execution Successful ===
38°C
Mostly sunny
ENG
IN

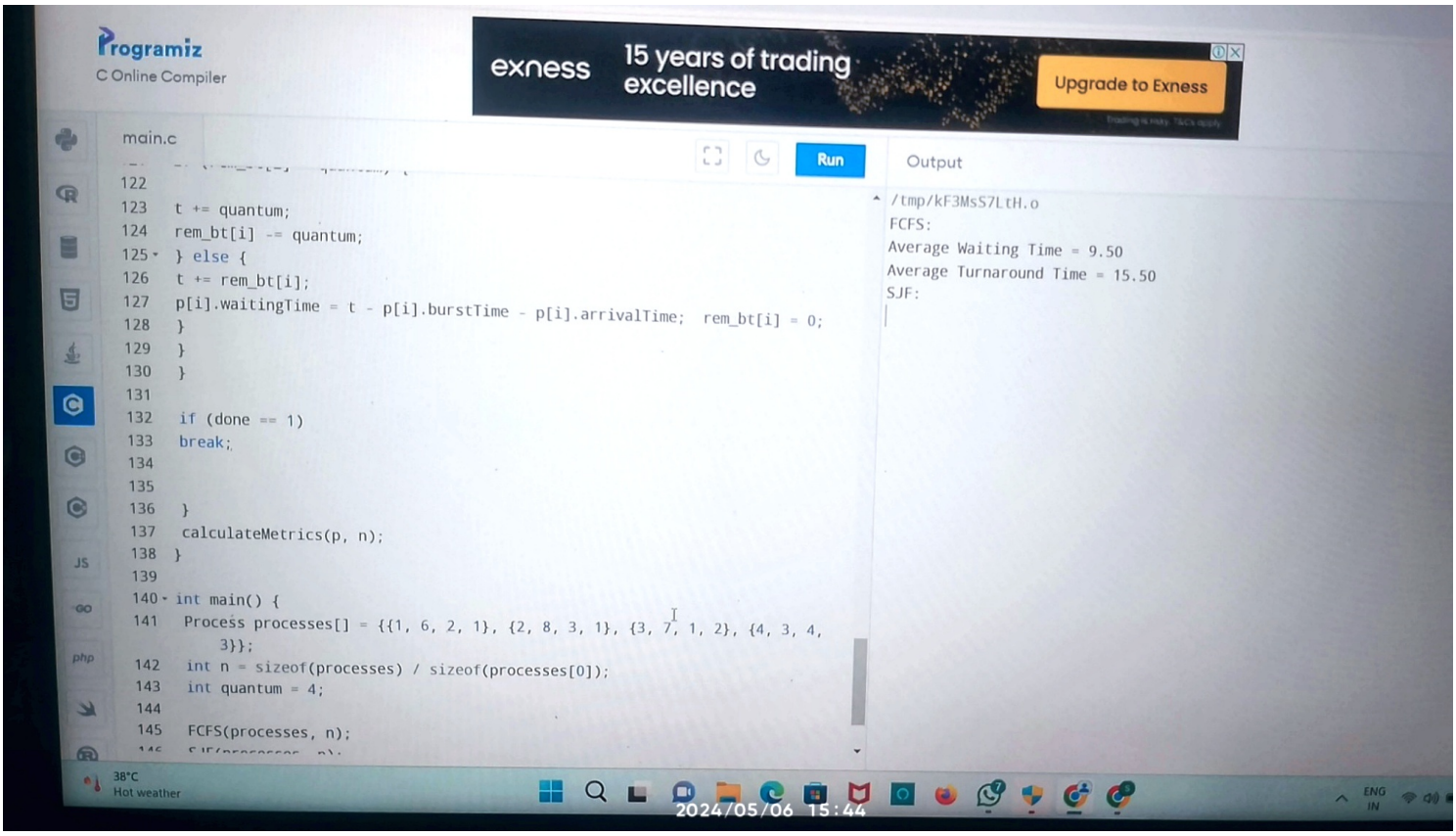

Programiz
C Online Compiler
Programiz PRO
Premium Coding
Courses by Programiz
Learn More
main.c
Run
Output
Programiz PRO
#include <stdio.h>
#include <stdlib.h>
typedef struct {
int id;
int burstTime;
int priority;
int arrivalTime;
int waitingTime;
int turnaroundTime;
} Process;
// Utility function to sort processes by arrival time
int compareArrival(const void *a, const void *b) {
Process *p1 = (Process *)a;
Process *p2 = (Process *)b;
return p1->arrivalTime - p2->arrivalTime;
}
// Utility function to sort processes by burst time for SJF
int compareBurstTime(const void *a, const void *b) {
Process *p1 = (Process *)a;
Process *p2 = (Process *)b;
return p1->burstTime - p2->burstTime;
}
// Utility function to sort processes by priority for Priority Scheduling
int comparePriority(const void *a, const void *b) {
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
BSE smlcap
-1.02%
ENG
IN
15:43
06-05-2024
2024/05/06 15:43

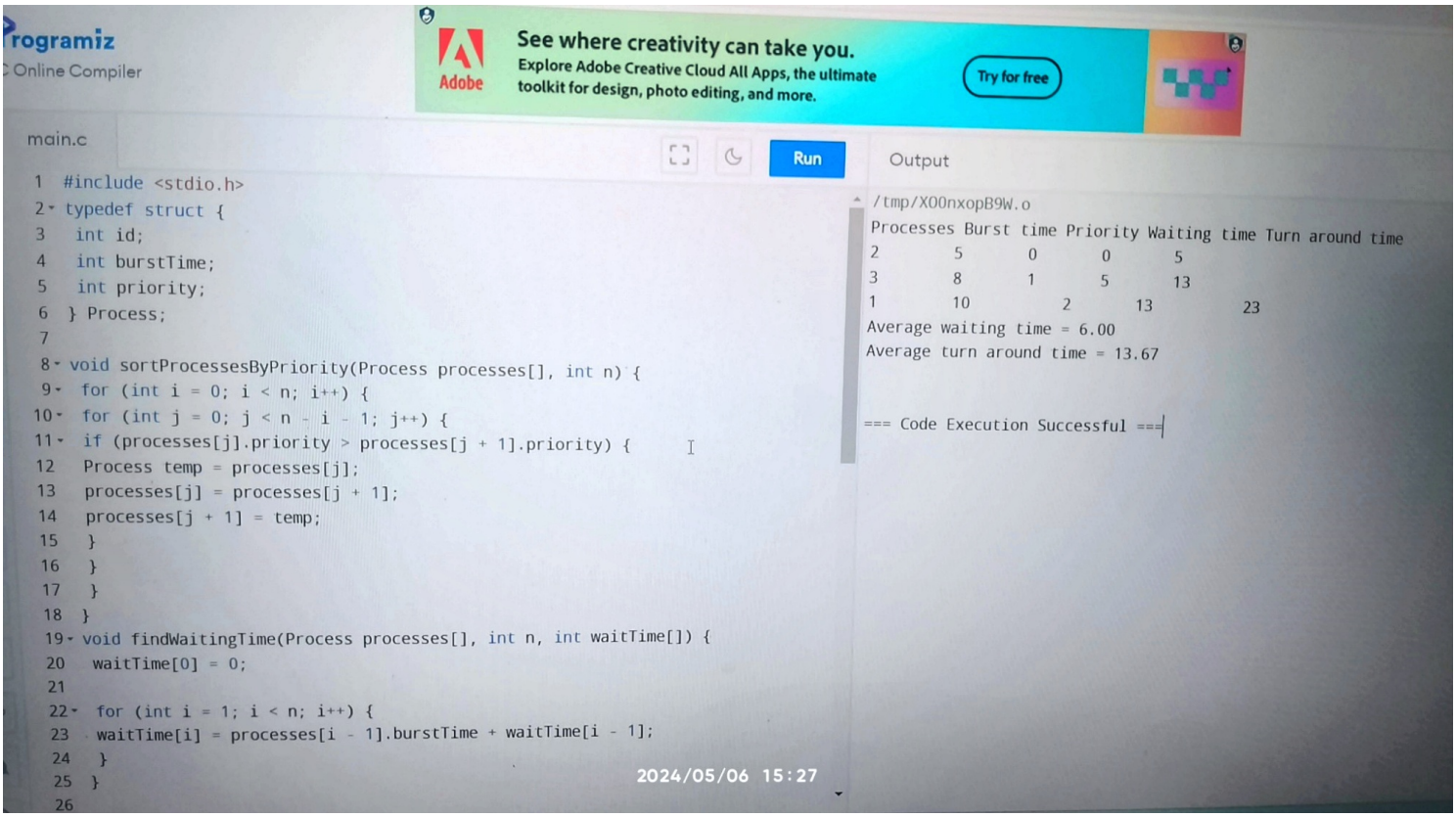

Programiz
C Online Compiler
Programiz PRO
Premium Coding
Courses by Programiz
Learn More
main.c
Run
Output
Programiz PRO
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
int comparePriority(const void *a, const void *b) {
Process *p1 = (Process *)a;
Process *p2 = (Process *)b;
return p1->priority - p2->priority;
}
// Function to calculate waiting time and turnaround time
void calculateMetrics(Process p[], int n) {
int total_wt = 0, total_tat = 0;
for (int i = 0; i < n; i++) {
p[i].turnaroundTime = p[i].burstTime + p[i].waitingTime;
total_wt += p[i].waitingTime;
total_tat += p[i].turnaroundTime;
}
printf("Average Waiting Time = %.2f\n", (float)total_wt / n);
printf("Average Turnaround Time = %.2f\n", (float)total_tat / n);
}
// First Come First Serve Algorithm
void FCFS(Process p[], int n) {
printf("FCFS:\n");
qsort(p, n, sizeof(Process), compareArrival);
int currentTime = 0;
for (int i = 0; i < n; i++) {
BSE smlcap
-1.02%
2024/05/06 15:43
ENG
IN
06-05-2024

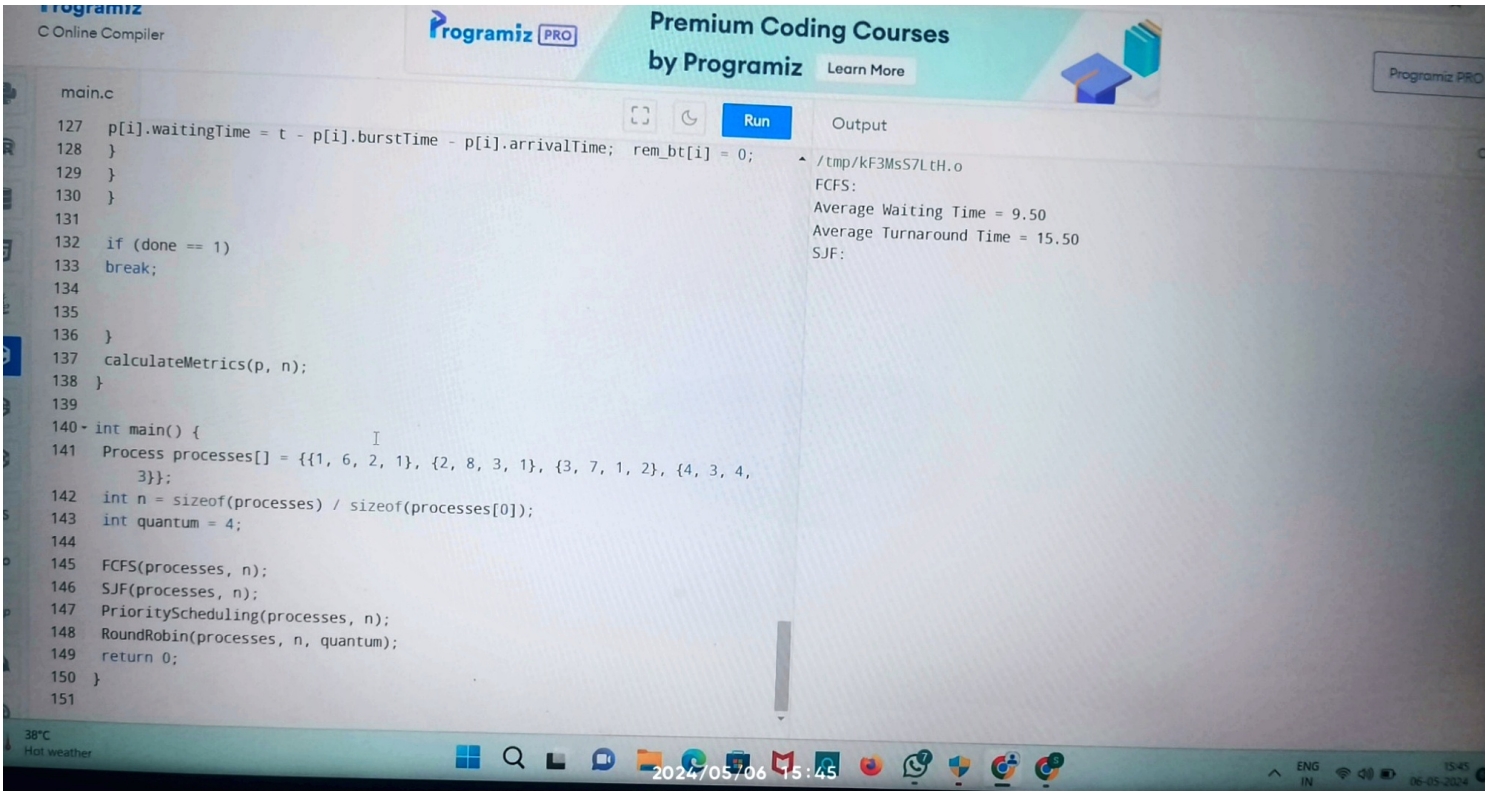

Programiz
C Online Compiler
Programiz PRO
Premium Coding Courses by Programiz
Learn More
main.c
Run
Output
for (int i = 0; i < n; i++) {
if (currentTime < p[i].arrivalTime)
currentTime = p[i].arrivalTime;
p[i].waitingTime = currentTime - p[i].arrivalTime; currentTime += p[i]
.burstTime;
}
calculateMetrics(p, n);
}
// Shortest Job First Algorithm (non-preemptive)
void SJF(Process p[], int n) {
printf("SJF:\n");
qsort(p, n, sizeof(Process), compareArrival);
int completed = 0, currentTime = 0, minIndex = -1;
int remaining = n;
while (completed < n) {
// Find process with shortest burst time among the arrived processes
int minBurst = 999999;
for (int i = 0; i < n; i++) {
if (p[i].arrivalTime <= currentTime && p[i].burstTime < minBurst && p[i]
.waitingTime == 0) { minBurst = p[i].burstTime;
minIndex = i;
}
}
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
38°C
Mostly sunny
ENG
IN
2024/05/06 15:43


Programiz
C Online Compiler
Programiz PRO
Premium Coding
Courses by Programiz
Learn More
main.c
Run
Output
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
73 }
74
75 if (minIndex == -1) {
76 currentTime++;
77 } else {
78 p[minIndex].waitingTime = currentTime - p[minIndex].arrivalTime;
79 currentTime += p[minIndex].burstTime;
80 completed++;
81 p[minIndex].waitingTime = (p[minIndex].waitingTime < 0) ? 0 : p[minIndex]
.waitingTime; minIndex = -1;
82 }
83 }
84
85 calculateMetrics(p, n);
86 }
87
88 // Priority Scheduling Algorithm (non-preemptive)
89
90 void PriorityScheduling(Process p[], int n) {
91 printf("Priority:\n");
92 qsort(p, n, sizeof(Process), comparePriority);
93
94 int currentTime = 0;
95 for (int i = 0; i < n; i++) {
96 if (currentTime < p[i].arrivalTime)
2024/05/06 15:43
38°C


Online C++ Compiler
Online C Compiler
Online C Compiler
Online C Compiler
Online C Compiler
Online C Compiler
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
Programiz PRO
Premium Coding Courses
by Programiz
Learn More
main.c
Run
Output
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
96 if (currentTime < p[i].arrivalTime)
97 currentTime = p[i].arrivalTime;
98 p[i].waitingTime = currentTime - p[i].arrivalTime;
99 currentTime += p[i].burstTime;
100 }
101
102 calculateMetrics(p, n);
103 }
104
105 // Round Robin Algorithm
106 void RoundRobin(Process p[], int n, int quantum) {
107 printf("Round Robin:\n");
108 qsort(p, n, sizeof(Process), compareArrival);
109
110 int rem_bt[n];
111 for (int i = 0; i < n; i++) {
112 rem_bt[i] = p[i].burstTime;
113 }
114
115 int t = 0;
116 while (1) {
117 int done = 1;
118 for (int i = 0; i < n; i++) {
119 if (rem_bt[i] > 0) {
120 done = 0;
121 if (rem_bt[i] > quantum) {
38°C
Mostly sunny
2024/05/06 15:44
ENG
IN


Programiz
C Online Compiler
exness
15 years of trading excellence
Upgrade to Exness
main.c
Run
Output
t += quantum;
rem_bt[i] -= quantum;
} else {
t += rem_bt[i];
p[i].waitingTime = t - p[i].burstTime - p[i].arrivalTime; rem_bt[i] = 0;
}
}
}
if (done == 1)
break;
}
calculateMetrics(p, n);
}
int main() {
Process processes[] = {{1, 6, 2, 1}, {2, 8, 3, 1}, {3, 7, 1, 2}, {4, 3, 4, 3}};
int n = sizeof(processes) / sizeof(processes[0]);
int quantum = 4;
FCFS(processes, n);
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
38°C
Hot weather
2024/05/06 15:44
ENG
IN


rogramiz
Online Compiler
Adobe
See where creativity can take you.
Explore Adobe Creative Cloud All Apps, the ultimate toolkit for design, photo editing, and more.
Try for free
main.c
Run
Output
#include <stdio.h>
typedef struct {
int id;
int burstTime;
int priority;
} Process;
void sortProcessesByPriority(Process processes[], int n) {
for (int i = 0; i < n; i++) {
for (int j = 0; j < n - i - 1; j++) {
if (processes[j].priority > processes[j + 1].priority) {
Process temp = processes[j];
processes[j] = processes[j + 1];
processes[j + 1] = temp;
}
}
}
}
void findWaitingTime(Process processes[], int n, int waitTime[]) {
waitTime[0] = 0;
for (int i = 1; i < n; i++) {
waitTime[i] = processes[i - 1].burstTime + waitTime[i - 1];
}
}
/tmp/XO0nxopB9W.o
Processes Burst time Priority Waiting time Turn around time
2 5 0 0 5
3 8 1 5 13
1 10 2 13 23
Average waiting time = 6.00
Average turn around time = 13.67
=== Code Execution Successful ===
2024/05/06 15:27


C Online Compiler
Programiz PRO
Premium Coding Courses
by Programiz
Learn More
main.c
Run
Output
p[i].waitingTime = t - p[i].burstTime - p[i].arrivalTime; rem_bt[i] = 0;
}
}
}
if (done == 1)
break;
}
calculateMetrics(p, n);
}
int main() {
Process processes[] = {{1, 6, 2, 1}, {2, 8, 3, 1}, {3, 7, 1, 2}, {4, 3, 4, 3}};
int n = sizeof(processes) / sizeof(processes[0]);
int quantum = 4;
FCFS(processes, n);
SJF(processes, n);
PriorityScheduling(processes, n);
RoundRobin(processes, n, quantum);
return 0;
}
/tmp/kF3MsS7LtH.o
FCFS:
Average Waiting Time = 9.50
Average Turnaround Time = 15.50
SJF:
38°C
Hot weather
2024/05/06 15:45