**Name: Srinivas Sanjeevkumar Chenna**

**Division: A**

**Roll Number: 48**

**Subject: Operating System (OS) LAB**

**Lab Assignment 3**

**Implement following CPU scheduling algorithms**

1. FCFS

#include <stdio.h>

int main()

{

int n, at[20], bt[20], wt[20], tt[20], ft[20], i, j, sum = 0;

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

scanf("%d", &n);

printf("\nEnter Process Arrival Time -\n");

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

{

printf("P[%d]: ", i + 1);

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

}

printf("\nEnter Process Burst Time -\n");

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

{

printf("P[%d]: ", i + 1);

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

}

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

{

sum = 0;

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

{

sum += bt[j];

}

ft[i] = sum;

}

printf("\*\*\*\*\*\*\*\*\*\*Gnatt Chart\*\*\*\*\*\*\*\*\*\*");

printf("\n\n");

printf("|");

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

{

printf(" P%d |",i+1);

}

printf("\n");

printf("0");

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

{

printf(" %d",ft[i]);

}

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

{

tt[i] = ft[i] - at[i];

}

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

{

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

}

printf("\nProcess\tArrival Time\tBurst Time\tFinal Time\tTurnaround Time\tWaiting Time\n");

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

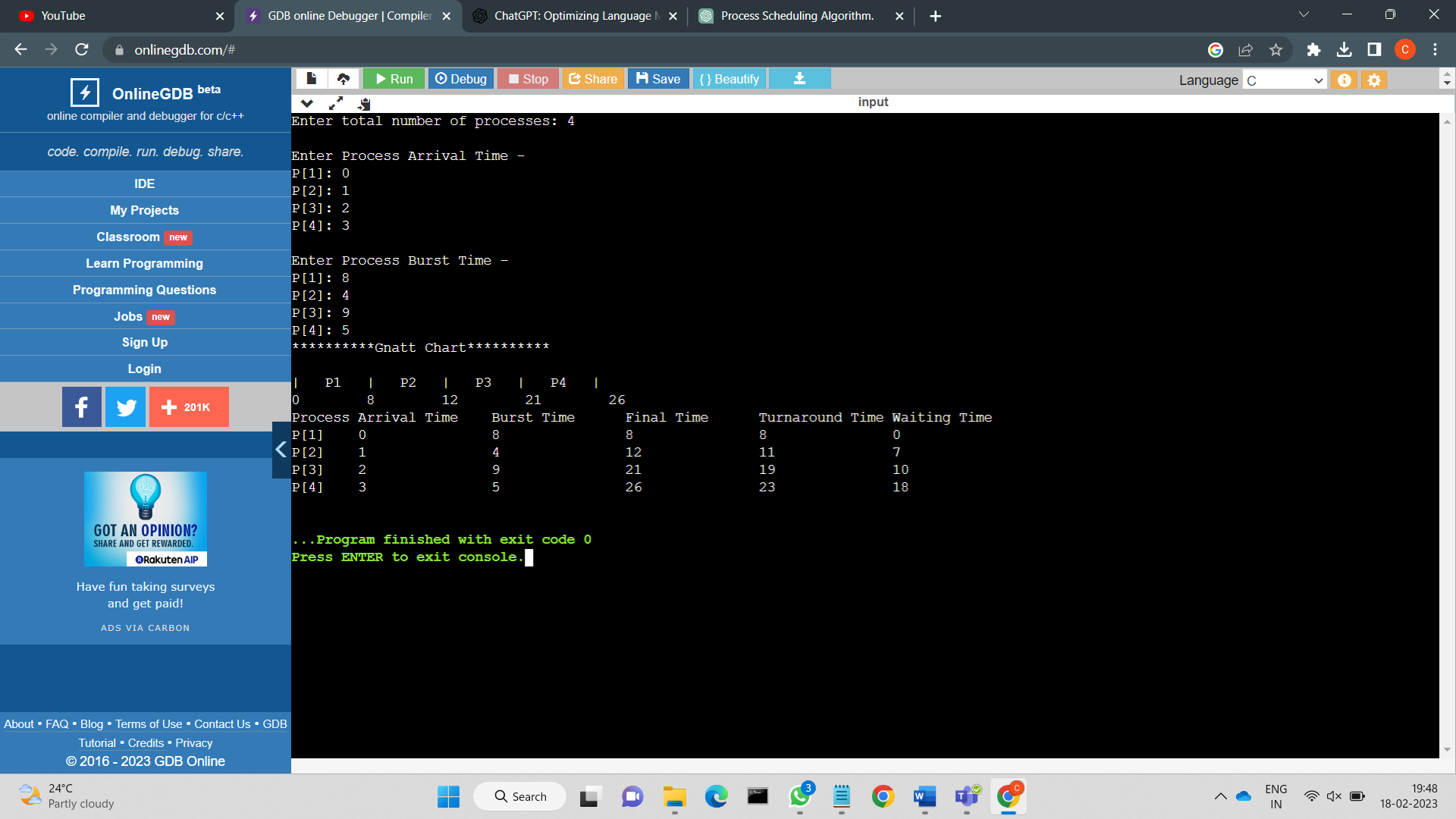
{

printf("P[%d]\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, at[i], bt[i], ft[i], tt[i], wt[i]);

}

return 0;

}



2. SJF (preemptive )

#include <stdio.h>

#include <stdlib.h>

struct process

{

int id, AT, BT, FT, TT, WT, RBT;

};

void SJF\_P(struct process pr[], int size)

{

int current\_time = 0, total\_WT = 0, total\_TT = 0;

int remaining\_processes = size;

printf("The Gantt Chart is:\n");

while (remaining\_processes)

{ int shortest\_job\_index = -1;

int shortest\_job\_burst\_time = 99999;

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

{

if (pr[i].RBT > 0 && pr[i].AT <= current\_time)

{

if (pr[i].RBT < shortest\_job\_burst\_time)

{

shortest\_job\_index = i;

shortest\_job\_burst\_time = pr[i].RBT;

}

}

}

if (shortest\_job\_index == -1)

{

current\_time++;

continue;

}

pr[shortest\_job\_index].RBT--;

current\_time++;

printf("P%d(%d) |", pr[shortest\_job\_index].id, current\_time);

if (pr[shortest\_job\_index].RBT == 0)

{

pr[shortest\_job\_index].TT = current\_time - pr[shortest\_job\_index].AT;

pr[shortest\_job\_index].WT = pr[shortest\_job\_index].TT - pr[shortest\_job\_index].BT;

pr[shortest\_job\_index].FT = pr[shortest\_job\_index].TT + pr[shortest\_job\_index].AT;

remaining\_processes--;

}

}

printf("\n\n");

printf("Process id\tArrival time\tBurst time\tFinish time\tTurnaround time\t Waiting time\n");

for (int p = 0; p < size; p++)

{

printf("%d\t\t", pr[p].id);

printf("%d\t\t", pr[p].AT);

printf(" %d\t\t", pr[p].BT);

printf("%d\t\t", pr[p].FT);

printf("%d \t\t", pr[p].TT);

printf(" %d\t ", pr[p].WT);

printf("\n");

}

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

{

total\_WT = total\_WT + pr[i].WT;

total\_TT = total\_TT + pr[i].TT;

}

float avg\_WT = (float)total\_WT / size;

float avg\_TT = (float)total\_TT / size;

printf("Average Waiting Time: %f\n", avg\_WT);

printf("Average TurnAround Time: %f", avg\_TT);

}

int main()

{

int n;

printf("Enter no of processes : ");

scanf("%d", &n);

struct process p[n];

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

{

printf("Enter process number : ");

scanf("%d", &p[i].id);

printf("Enter AT : ");

scanf("%d", &p[i].AT);

printf("Enter BT : ");

scanf("%d", &p[i].BT);

p[i].RBT = p[i].BT;

}

SJF\_P(p, n);

return 0;

}

A screenshot of a computer

Description automatically generated