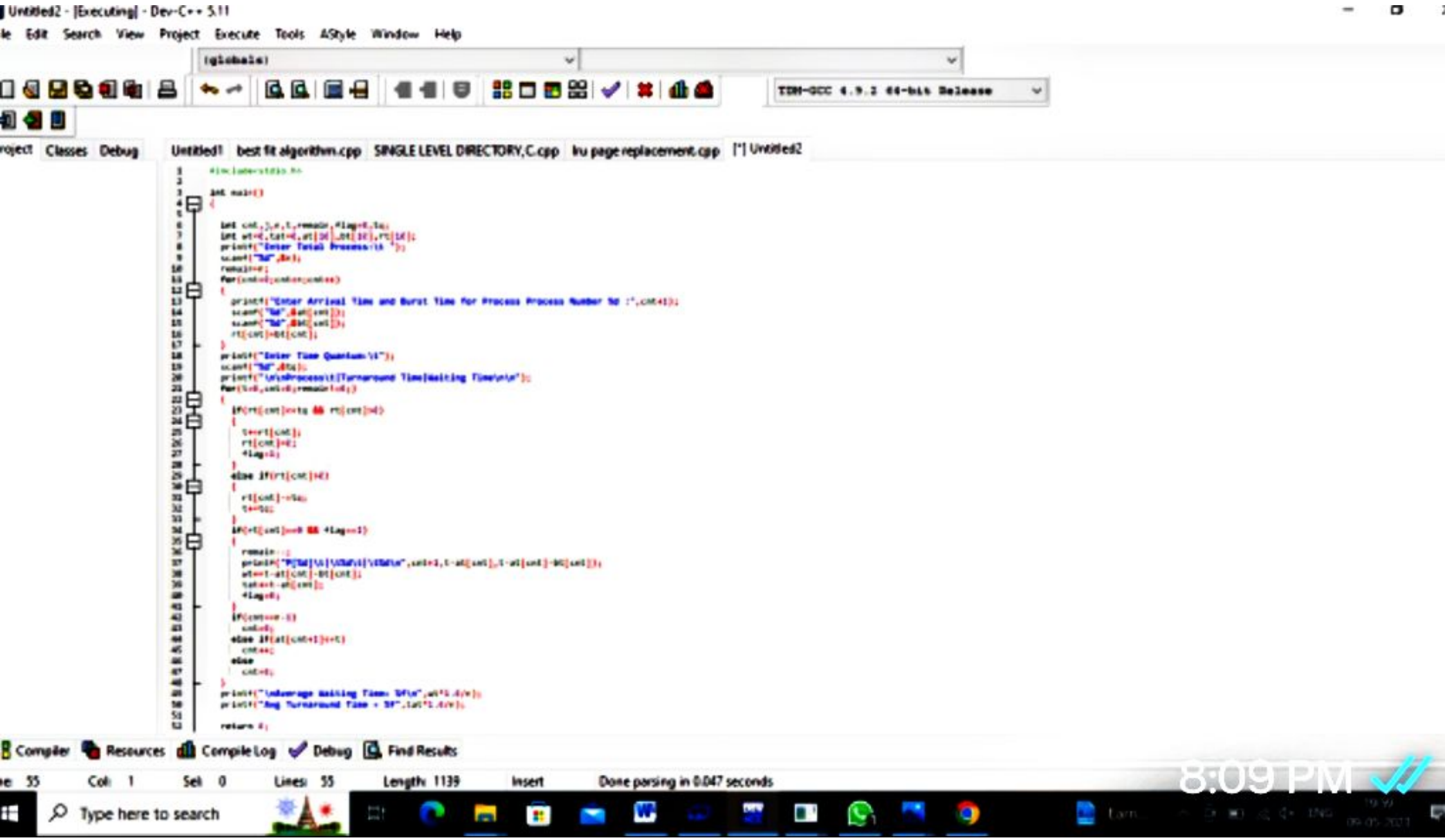
17. Write a program to compute the average waiting time and average turnaround time based on Round Robin scheduling for the following process with the given CPU burst times and quantum time slots 4 ms, ( and the assumption that all jobs arrive at the same time.)

| Process | Burst Time |
| --- | --- |
| P124 | |
| P23 | |
| P33 | |


Untitled2 - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
TDM-GCC 4.9.2 64-bit Release
Project Classes Debug
Untitled1
best fit algorithm.cpp
SINGLE LEVEL DIRECTORY,C.cpp
lru page replacement.cpp
[*] Untitled2
#include<stdio.h>
int main()
{
int cnt,j,n,t,remain,flag=0,tq;
int wt=0,tat=0,at[10],bt[10],rt[10];
printf("Enter Total Process:\t ");
scanf("%d",&n);
remain=n;
for(cnt=0;cnt<n;cnt++)
{
printf("Enter Arrival Time and Burst Time for Process Process Number %d :",cnt+1);
scanf("%d",&at[cnt]);
scanf("%d",&bt[cnt]);
rt[cnt]=bt[cnt];
}
printf("Enter Time Quantum:\t");
scanf("%d",&tq);
printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(t=0,cnt=0;remain!=0;)
{
if(rt[cnt]<=tq && rt[cnt]>0)
{
t+=rt[cnt];
rt[cnt]=0;
flag=1;
}
else if(rt[cnt]>0)
{
rt[cnt]-=tq;
t+=tq;
}
if(rt[cnt]==0 && flag==1)
{
remain--;
printf("P[%d]\t|\t%d\t|\t%d\n",cnt+1,t-at[cnt],t-at[cnt]-bt[cnt]);
wt+=t-at[cnt]-bt[cnt];
tat+=t-at[cnt];
flag=0;
}
if(cnt==n-1)
cnt=0;
else if(at[cnt+1]<=t)
cnt++;
else
cnt=0;
}
printf("\nAverage Waiting Time= %f\n",wt*1.0/n);
printf("Avg Turnaround Time = %f",tat*1.0/n);
return 0;
}
Compiler Resources Compile Log Debug Find Results
Line: 55 Col: 1 Sel: 0 Lines: 55 Length: 1139 Insert Done parsing in 0.047 seconds
Type here to search
8:09 PM

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

TDM-GCC 4.9.2 64-bit Release

Project Classes Debug

Untitled1 | best fit algorithm.cpp | SINGLE LEVEL DIRECTORY,C.cpp | lru page replacement.cpp | ROUND ROBIN ALGORITHM IN C.cpp | [*] Untitled3

```c
#include<stdio.h>

#define MAX 100

struct Process {
    int burstTime;
    int remainingTime;
    int waitingTime;
    int turnaroundTime;
};

void calculateWaitingTime(struct Process processes[], int n, int quantum) {
    int remainingBurstTime[MAX];
    for (int i = 0; i < n; i++)
        remainingBurstTime[i] = processes[i].burstTime;

    int currentTime = 0;
    while (1) {
        int done = 1;
        for (int i = 0; i < n; i++) {
            if (remainingBurstTime[i] > 0) {
                done = 0;
                if (remainingBurstTime[i] > quantum) {
                    currentTime += quantum;
                    remainingBurstTime[i] -= quantum;
                } else {
                    currentTime += remainingBurstTime[i];
                    processes[i].waitingTime = currentTime - processes[i].burstTime;
                    remainingBurstTime[i] = 0;
                }
            }
        }
        if (done)
            break;
    }
}

void calculateTurnaroundTime(struct Process processes[], int n) {
    for (int i = 0; i < n; i++)
        processes[i].turnaroundTime = processes[i].burstTime + processes[i].waitingTime;
}

void calculateAverageTime(struct Process processes[], int n, int quantum) {
    calculateWaitingTime(processes, n, quantum);
    calculateTurnaroundTime(processes, n);

    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

Compiler | Resources | Compile Log | Debug | Find Results

Line: 87 Col: 1 Sel: 0 Lines: 87 Length: 2595 Insert Done parsing in 0.046 seconds

Type here to search

30°C ENG 20:06 09-05-2023

Edit Search View Project Execute Tools AStyle Window Help

(globals)

TDM-GCC 4.9.2 64-bit Release

Classes Debug

Untitled1 | best fit algorithm.cpp | SINGLE LEVEL DIRECTORY,C.cpp | lru page replacement.cpp | ROUND ROBIN ALGORITHM IN C.cpp | [*] Untitled3

```
}

void calculateTurnaroundTime(struct Process processes[], int n) {
    for (int i = 0; i < n; i++)
        processes[i].turnaroundTime = processes[i].burstTime + processes[i].waitingTime;
}

void calculateAverageTime(struct Process processes[], int n, int quantum) {
    calculateWaitingTime(processes, n, quantum);
    calculateTurnaroundTime(processes, n);

    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        totalWaitingTime += processes[i].waitingTime;
        totalTurnaroundTime += processes[i].turnaroundTime;
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i+1, processes[i].burstTime, processes[i].waitingTime, processes[i].turnaroundTime);
    }

    float avgWaitingTime = (float)totalWaitingTime / n;
    float avgTurnaroundTime = (float)totalTurnaroundTime / n;

    printf("\nAverage Waiting Time: %.2f ms\n", avgWaitingTime);
    printf("Average Turnaround Time: %.2f ms\n", avgTurnaroundTime);
}

int main() {
    int n, quantum;

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

    struct Process processes[MAX];

    printf("Enter the burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        printf("P%d: ", i+1);
        scanf("%d", &processes[i].burstTime);
        processes[i].remainingTime = processes[i].burstTime;
    }

    printf("Enter the quantum time: ");
    scanf("%d", &quantum);

    calculateAverageTime(processes, n, quantum);

    return 0;
}
```

Compiler | Resources | Compile Log | Debug | Find Results

Ln: 87 Col: 1 Sel: 0 Lines: 87 Length: 2595 Insert Done parsing in 0.046 seconds

Type here to search

30°C ENG 20:07 09-05-2023

```
Enter the number of processes: 3
Enter the burst time for each process:
P1: 24
P2: 3
P3: 3
Enter the quantum time: 4
Process Burst Time      Waiting Time    Turnaround Time
P1              24              6               30
P2              3               4               7
P3              3               7               10

Average Waiting Time: 5.67 ms
Average Turnaround Time: 15.67 ms

--------------------------------
Process exited after 21.14 seconds with return value 0
Press any key to continue . . .
```