**OPERATING SYSTEM LAB TASK – 05**

**Name:** Kamisha Salim

**S.ID:** 11070

**QUESTION – 1**

**CODE:**

#include<stdio.h>

int main()

{

int i,j,k,n,bt[20],wt[20],tat[20],pri[20],p[20],t,max,ct[20],temp;

float wtavg,tatavg,tmp=0;

printf("Enter the no. of processes: ");

scanf("%d",&n);

printf("\n");

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

{

p[i]=i;

printf("Enter burst time and arrival time for process %d: ", i);

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

ct[i]=bt[i];

}

printf("\nEnter the size of time slice: ");

scanf("%d",&t);

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

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

if(pri[i]>pri[k]){

temp=p[i];

p[i]=p[k];

p[k]=temp;

temp=bt[k];

bt[i]=bt[k];

bt[k]=temp;

temp=pri[i];

pri[i]=pri[k];

pri[k]=temp;

}

max=bt[0];

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

if(max<bt[i])

max=bt[i];

for(j=0;j<(max/t)+1;j++)

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

if(bt[i]!=0)

{

if(bt[i]<=t)

{

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

tmp=tmp+bt[i];

bt[i]=0;

}

else

{

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

tmp=tmp+t;

}

}

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

wt[i]=tat[i]-ct[i];

tatavg+=tat[i];

wtavg+=wt[i];

}

printf("\n\tPROCESS\t\tARRIVAL TIME\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");

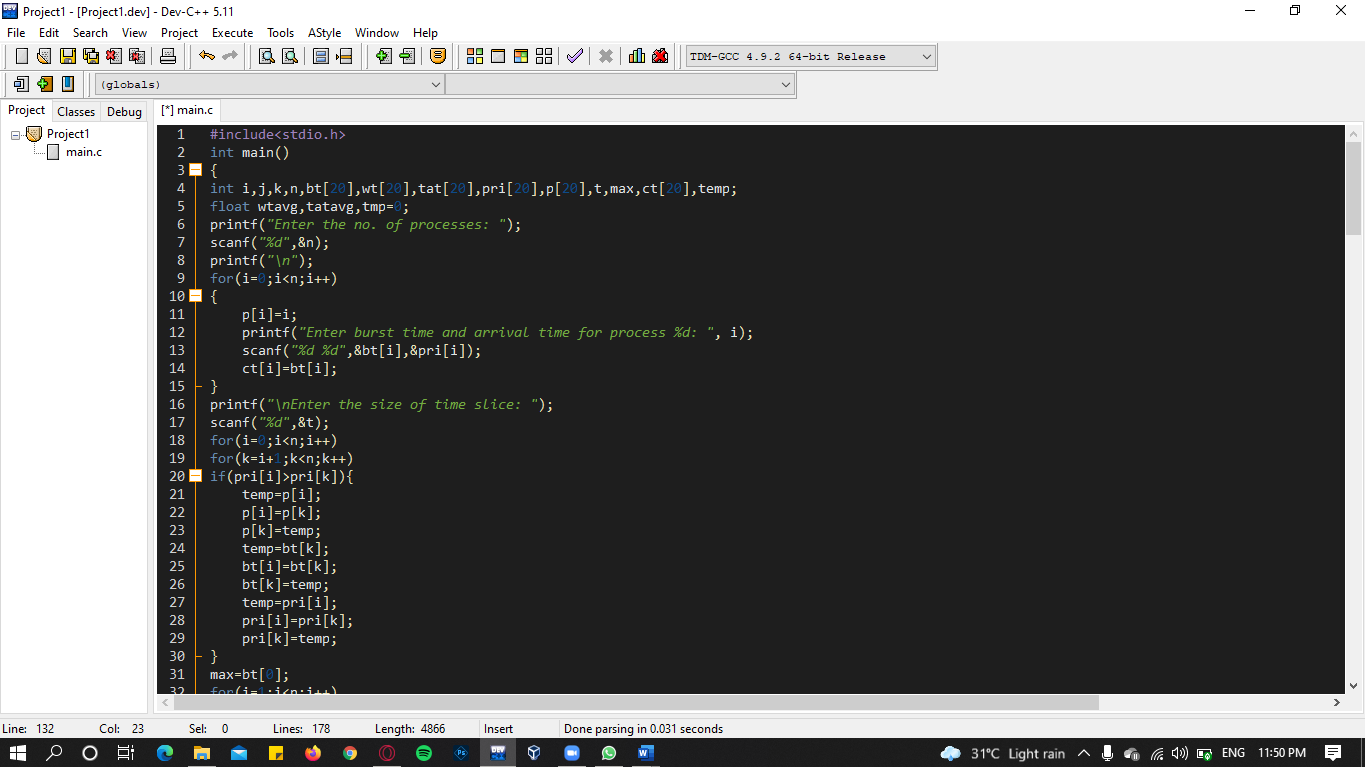
for(i=0;i<n;i++)

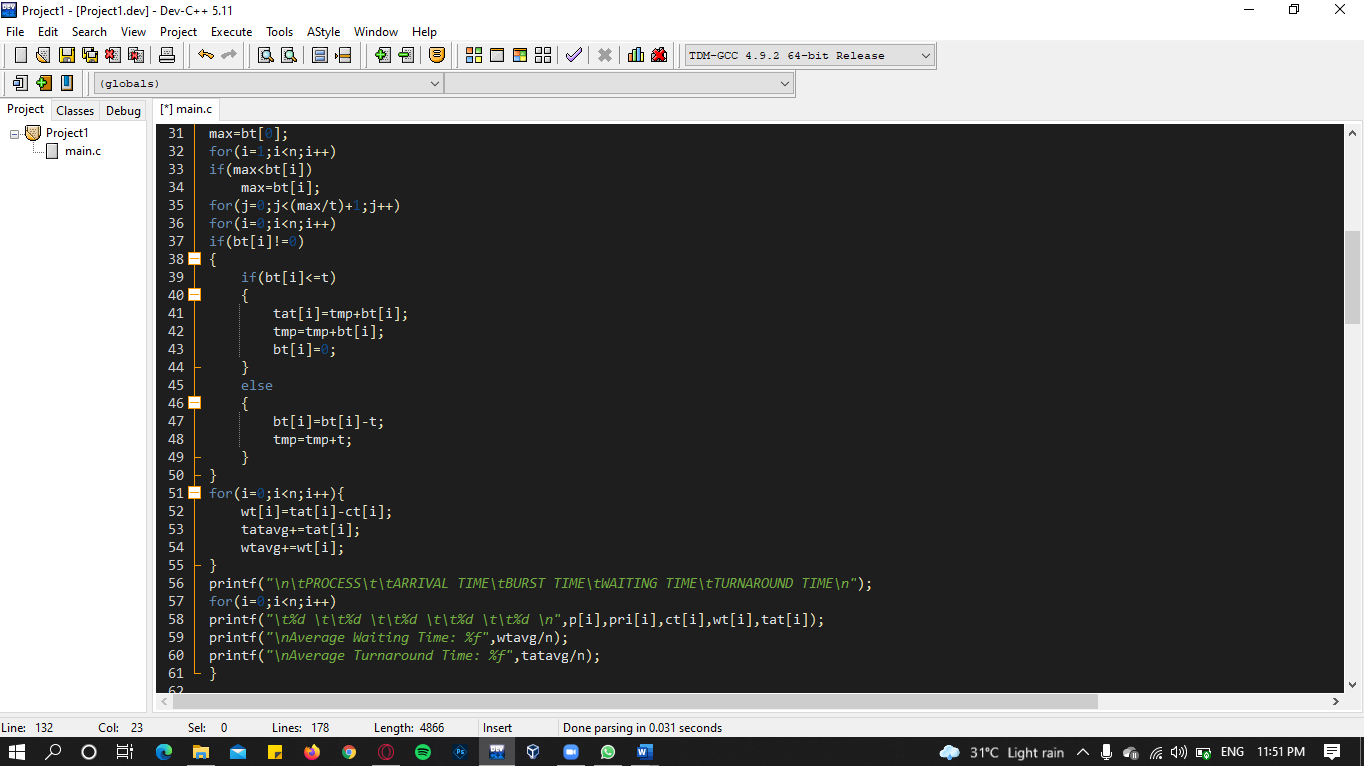
printf("\t%d \t\t%d \t\t%d \t\t%d \t\t%d \n",p[i],pri[i],ct[i],wt[i],tat[i]);

printf("\nAverage Waiting Time: %f",wtavg/n);

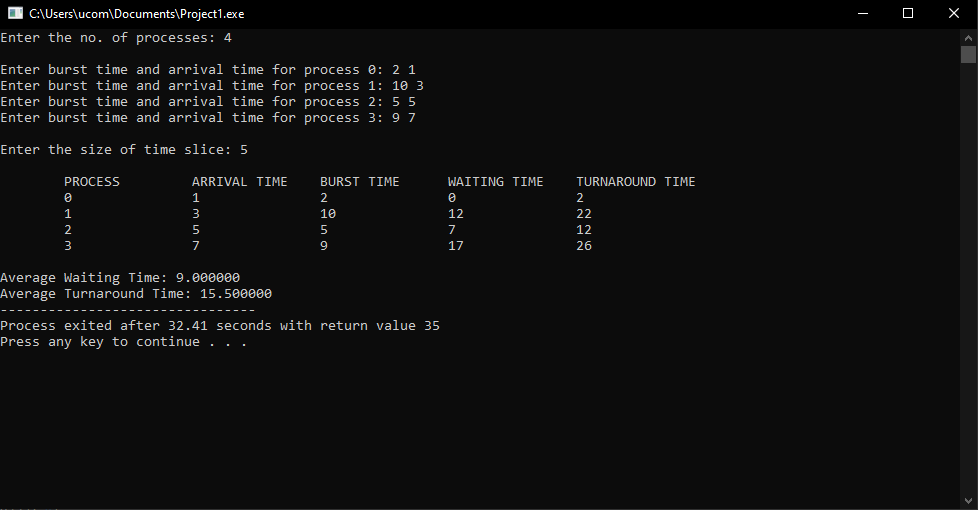
printf("\nAverage Turnaround Time: %f",tatavg/n);

}





**OUTPUT:**



**QUESTION – 2:**

Which of the following process scheduling algorithm may lead to starvation?

1. FIFO
2. Round Robin
3. Shortest Job Next
4. None of the above

**Answer:** c) Shortest Job Next

This is because if short processes are continually added, SJF will switch to the short ones and keep the processes which require a long time to complete in waiting, hence resulting in starvation of long processes.