OPERATING SYSTEM LAB TASK – 05

Name: Kamisha Salim

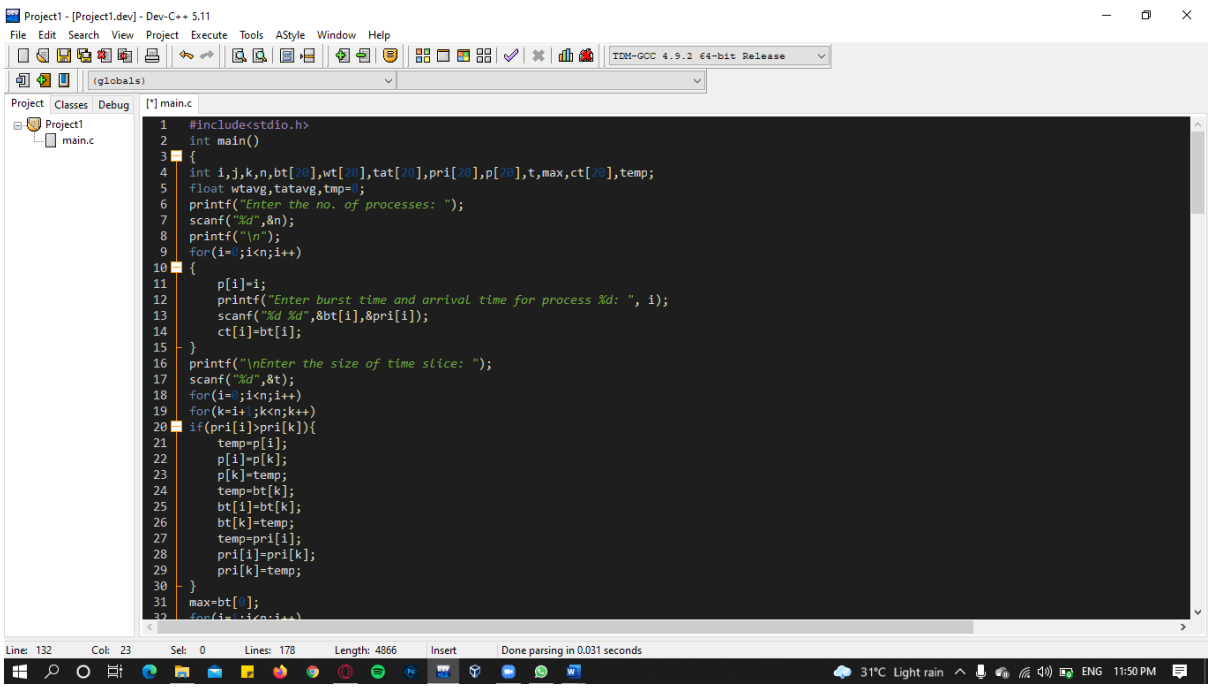
S.ID: 11070

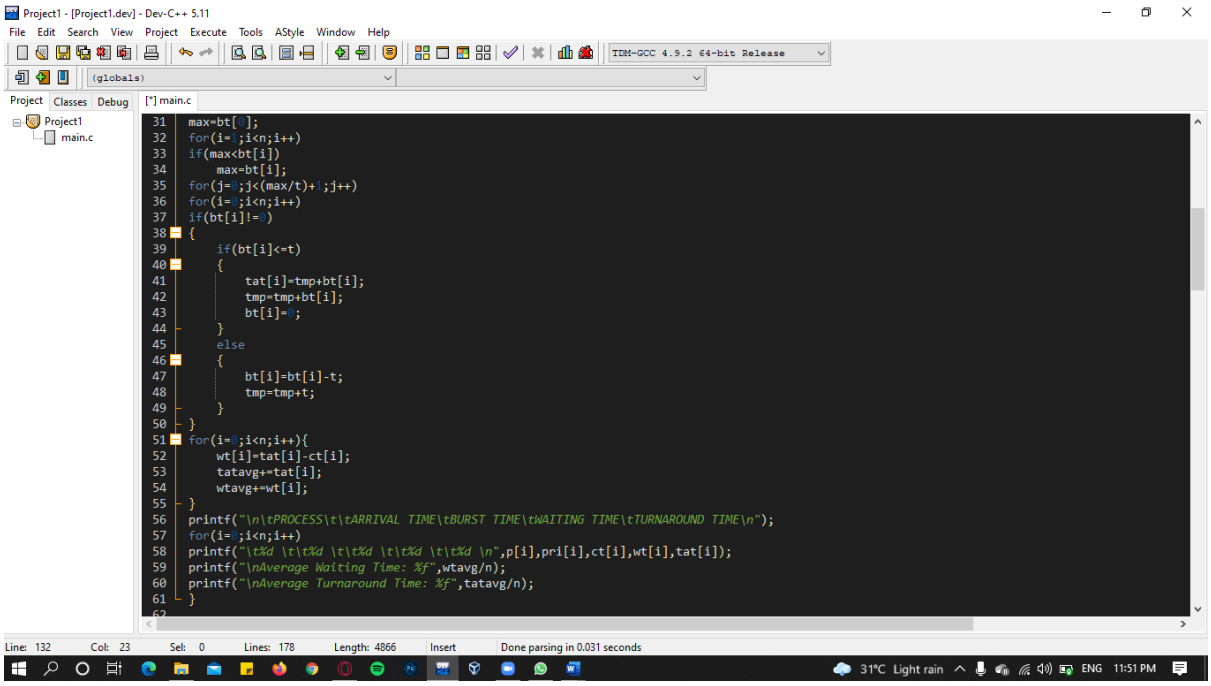
QUESTION – 1

CODE:

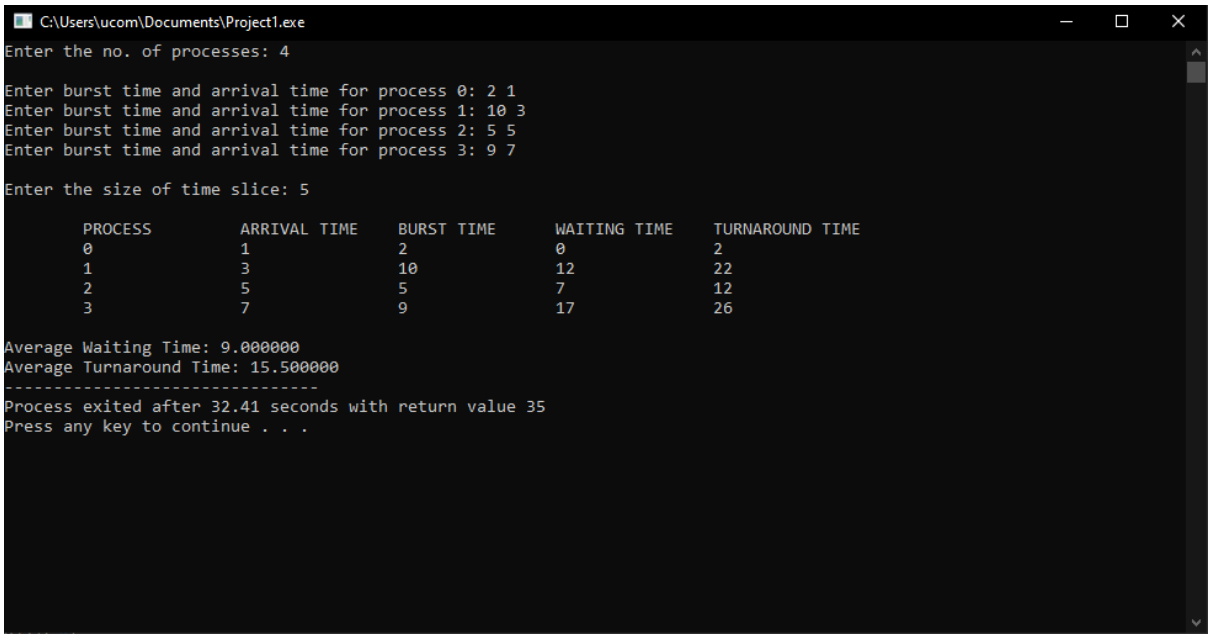
```
#include<stdio.h>
int main()
{
int i,j,k,n,bt[20],wt[20],tat[20],pri[20],p[20],t,max,ct[20],temp;
float wtavg,tatavg,tmp=0;
printf("Enter the no. of processes: ");
scanf("%d",&n);
printf("\n");
for(i=0;i<n;i++)
{
        p[i]=i;
        printf("Enter burst time and arrival time for process %d: ", 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);
}
```




Project1 - [Project1.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] main.c
Project1
main.c
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);
}
Line: 132 Col: 23 Sel: 0 Lines: 178 Length: 4866 Insert Done parsing in 0.031 seconds
31°C Light rain
ENG 11:51 PM

OUTPUT:


C:\Users\ucom\Documents\Project1.exe
Enter the no. of processes: 4
Enter burst time and arrival time for process 0: 2 1
Enter burst time and arrival time for process 1: 10 3
Enter burst time and arrival time for process 2: 5 5
Enter burst time and arrival time for process 3: 9 7
Enter the size of time slice: 5
PROCESS ARRIVAL TIME BURST TIME WAITING TIME TURNAROUND TIME
0 1 2 0 2
1 3 10 12 22
2 5 5 7 12
3 7 9 17 26
Average Waiting Time: 9.000000
Average Turnaround Time: 15.500000
Process exited after 32.41 seconds with return value 35
Press any key to continue . . .

QUESTION – 2:

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

a) FIFO
b) Round Robin
c) Shortest Job Next
d) 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.