OPERATING SYSTEM

ASSIGNMENT

Name-Nikhil Kumar

Student Id-11712912

Email Address-mnikhil1999@gmail.com

GitHub Link - https://github.com/mnikhil1999/Nikhil-Kumar.git

**Q1.** Write a program in C which reads input CPU bursts from a the first line of a text file named as CPU\_BURST.txt. Validate the input numbers whether the numbers are positive integers or not. Consider the numbers as CPU burst. If there are 5 positive integers in the first line of the text file then the program treat those argument as required CPU bust for P1, P2, P3, P4, and P5 process and calculate average waiting time and average turnaround time. Consider used scheduling algorithm as SJF and same arrival time for all the processes.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*fp = fopen("cpu\_burst.txt", "r");

int bt[20],p[20],wt[20],tat[20],i=0,j,n=5,total=0,pos,temp;

float avg\_wt,avg\_tat;

printf("\nReading CPU\_BURST.txt File\n");

//for(i=0;i<5;i++)

while((getc(fp))!=EOF)

{

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

if(bt[i]>0){

p[i]=i+1; i++;} //contains process number

}

n=i;

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

{

pos=i;

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

{

if(bt[j]<bt[pos])

pos=j;

}

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0; //waiting time for first process will be zero

//calculate waiting time

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n; //average waiting time

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

tat[i]=bt[i]+wt[i]; //calculate turnaround time

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=(float)total/n; //average turnaround time

printf("\n\nAverage Waiting Time=%f",avg\_wt);

printf("\nAverage Turnaround Time=%f\n",avg\_tat);

fclose(fp);

return 0;

}

Note- For succesfully running of the program we have to add an extra file cpu\_brust to folder where program is saved,so the program can read the values and it also don’t take the 0th place value of file so we have to give file 6 values.

1. Explain the problem in terms of operating system concept?

Answer-5 process are running consequtively with the help of Shortest Job First(SJF).We have to calculate the average waiting time and average turnaround time.In this problem arrival time is equal for all so the shortest job will be find out with the help of burst time..

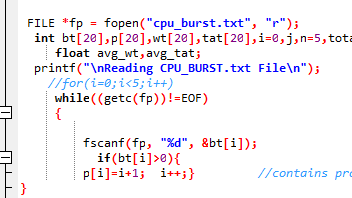
1. Write the algorithm for proposed solution of the assigned problem.

Answer-We have used SJF(Shortest Job First). **Shortest job first** (**SJF**) or shortest job next, is a **scheduling** policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm. **Shortest Job first** has the advantage of having minimum average waiting time among all **scheduling** algorithms.

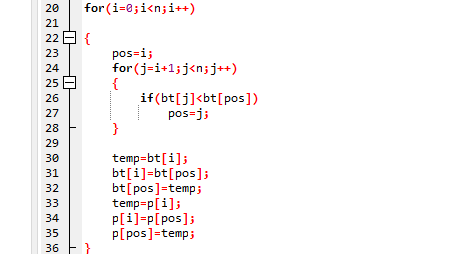
3. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint

Answer-

1.This is for reading of values from the file CPU\_BRUST



2.Waiting for the starting process to complete



4. Calculate complexity of implemented algorithm.

Answer-

**Complexity is O(n).**

5. Explain the boundary conditions of the implemented code.

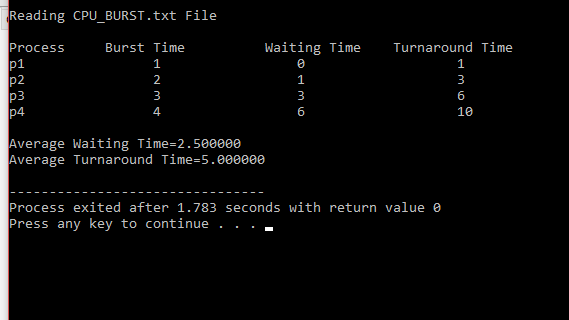
Answer-

* If the value is not positive,than the program will calculate the brust time and average waiting time without including that process
* Minimum 6 values should be provided in file.
* Arrival time is same for all process,so it all depends upon brust time
* If we provide more than 6 operation than it will include more than 5 process for calculating brust time and average waiting time.

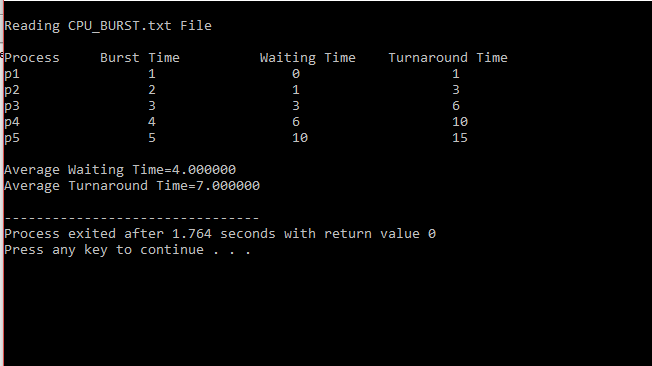
6. Explain all the test cases applied on the solution of assigned problem?

Answer-

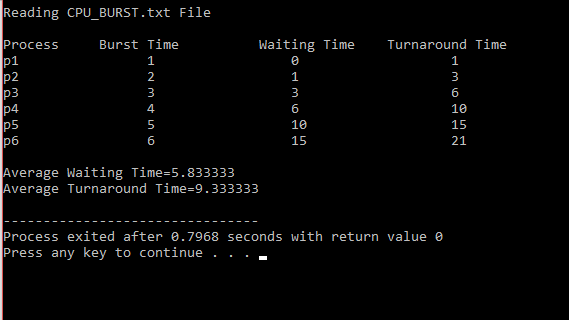
* Using only 4 process

****

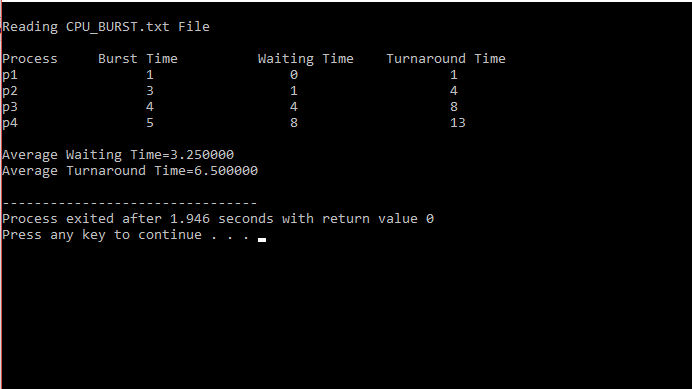
* Using 5 process

****

* Using 6 process

****

* Using one process out of five as negative

****

**Q2.** ASSIGNMENT COMPLETION PROBLEM

There are 3 student processes and 1 teacher process. Students are supposed to do their assignments and they need 3 things for that-pen, paper and question paper. The teacher has an infinite supply of all the three things. One students has pen, another has paper and another has question paper. The teacher places two things on a shared table and the student having the third complementary thing makes the assignment and tells the teacher on completion. The teacher then places an other two things out of the three and again the student having the third thing makes the assignment and tells the teacher on completion. This cycle continues. WAP to synchronise the teacher and the students

Code- #include<stdio.h>

#include<stdbool.h>

struct requirement

{

bool pen ;

bool paper ;

bool question\_paper ;

bool all\_three ;

};

int main()

{

int n=3;

struct requirement s[n];

s[0].pen=true;

s[0].paper = false;

s[0].question\_paper = false;

s[0].all\_three= false;

s[1].pen=false;

s[1].paper = true;

s[1].question\_paper = false;

s[1].all\_three = false;

s[2].pen=false;

s[2].paper = false;

s[2].question\_paper = true;

s[2].all\_three = false ;

while(s[0].all\_three==false||s[1].all\_three==false||s[2].all\_three==false)

{

int ch1,ch2;

printf("\nResources:\n1.pen\n2.paper\n3.question paper\n Enter the two things which is to be placed on the shared table: ");

scanf("%d%d",&ch1,&ch2);

if(ch1==1 && ch2==2 && s[2].all\_three==false)

{

s[2].all\_three=true ;

printf("Third Student has completed the task\n");

}

if(ch1==2 && ch2==3 && s[0].all\_three==false)

{

s[0].all\_three=true;

printf("First Student has completed the task\n");

}

if(ch1==1 && ch2==3 && s[1].all\_three==false)

{

s[1].all\_three=true;

printf("Second Student has completed the task\n");

}

}

printf("All the students now have completed their respective tasks succesfully\n");

return 0;

}

1. Explain the problem in terms of operating system concept?

**Description:** In this there are four process, 3 of them have one-one item. But the 4th have unlimited supply of all. 3 of them is asking for supply of the item form the fourth item.After completion of supply for one process it sends a message to the fourth item ,that supply has been completed

2. Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

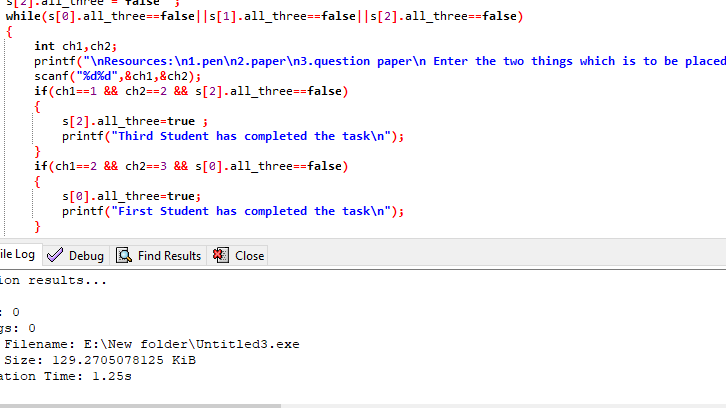
**Answer –**

**The complexity is O(n)**

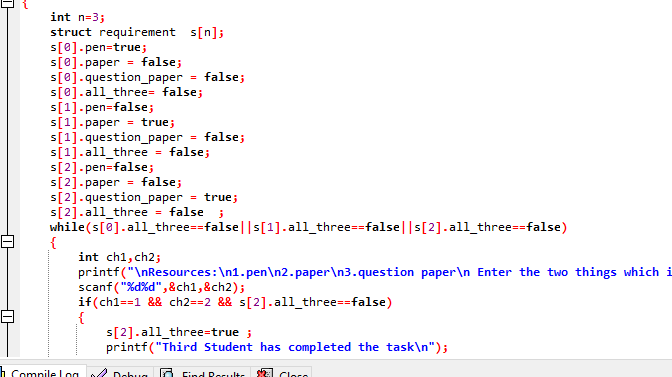
3. Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

**Code snippet:**

**1.**Asking for the items to be needed

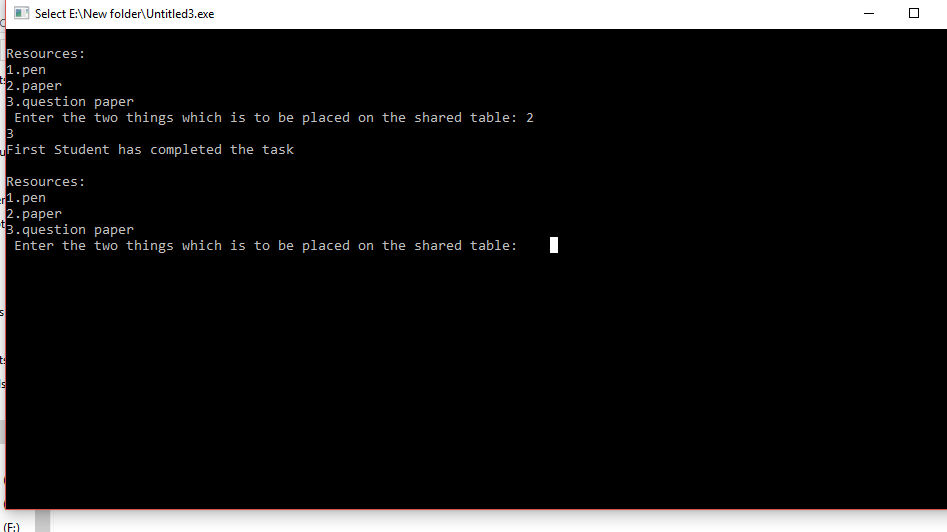


2.Initalizing the Boolean variable

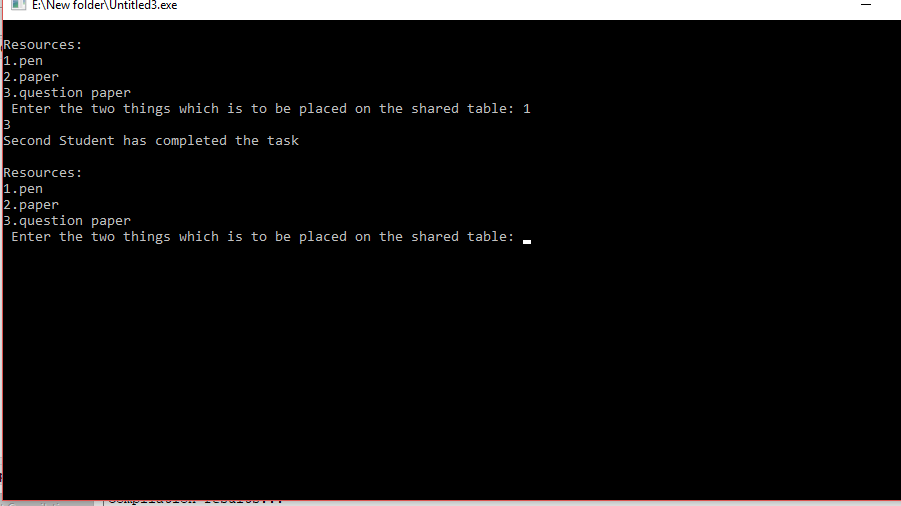


4. Explain all the test cases applied on the solution of assigned problem**.**

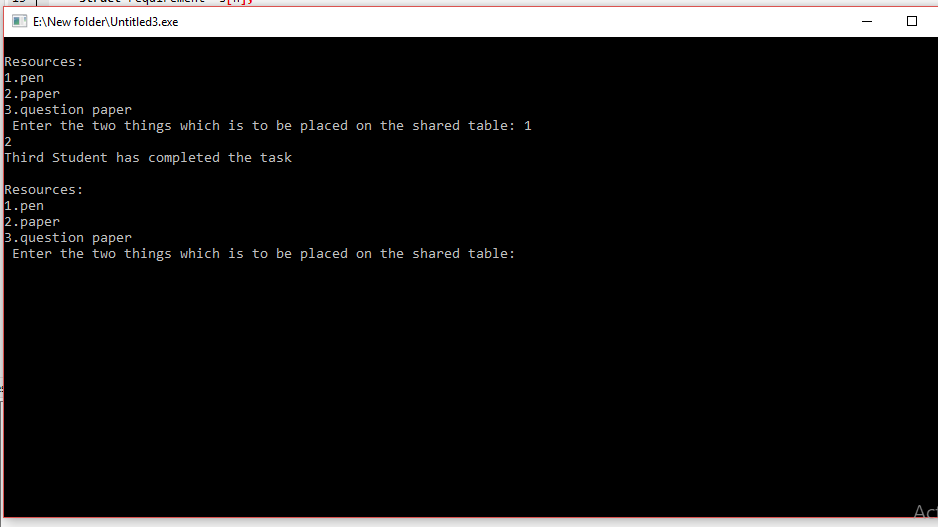
* When 1st process gets completed

****

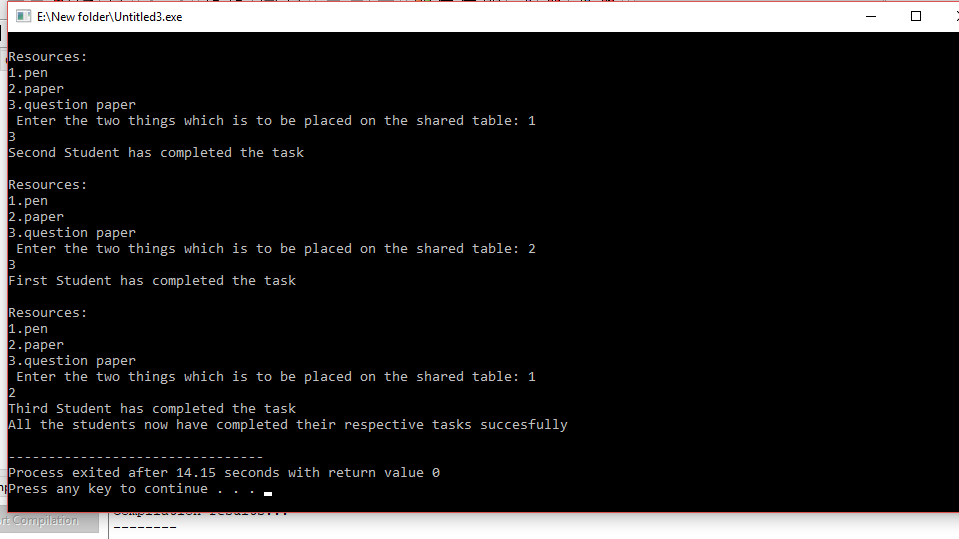
* When process 2nd gets completed

****

* When process 3rd gets completed

****

* When all three process gets completed

****