OPERATING SYSTEM LABORATORY MANUAL

UNIVERSITY OF THE PUNJAB

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY, LAHORE

DEPARTMENT OF COMPUTER SCIENCE

Course:	**Operating System Lab**	**Date:**
Course Code:	**CC-217-3L**	**Max Marks: 40**
Faculty/Instructor's Name & Email:	**Dr. Ahmad Hassan Butt (ahmad.hassan@pucit.edu.pk)**	

LAB MANUAL # 7

(SPRING 2023)

Name:_____ Enroll No: _____

Objective(s) :

To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE. To write a C program to implement the CPU scheduling algorithm for Shortest job first.

Lab Tasks :

Task 1 : Calculate the Average Time using FCFS Algorithm.

Task 2: Write the output of the program for First Come First Serve.

Task 3 : Calculate the Average Time using SJF Algorithm.

Task 4 : Write the output a program for Shortest Job First.

Lab Grading Sheet :

Task	Max Marks	Obtained Marks	Comments(*if any*)
1.	10		
2.	10		
3.	10		
4.	10		
Total	**40**		**Signature**

Note : Attempt all tasks and get them checked by your Instructor

Lab 7: Scheduling

Objective(s):

- To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE.
- To write a C program to implement the CPU scheduling algorithm for Shortest job first.

Tool(s) used:

Ubuntu, VIM Editor

First Come First Serve

CPU scheduler will decide which process should be given to the CPU for its execution. For this it uses different algorithms to choose among the process. One among that algorithm is FCFS algorithm. In this algorithm, the process which arrive first is given to the CPU after finishing its request only it will allow CPU to execute other process.

Task 1 : Calculate the Average Time using FCFS Algorithm.

Process	Duration	Order	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Task 2 : Write the output of the program for First Come First Serve.

Step 1 : Create the number of process.

Step 2 : Get the ID and Service time for each process.

Step 3 : Initially, Waiting time of first process is zero and Total time for the first process is the starting time of that process.

Step 4 : Calculate the Total time and Processing time for the remaining processes.

Step 5 : Waiting time of one process is the Total time of the previous process.

Step 6 : Total time of process is calculated by adding Waiting time and Service time.

Step 7 : Total waiting time is calculated by adding the waiting time for lack process.

Step 8 : Total turnaround time is calculated by adding all total time of each process.

Step 9 : Calculate Average waiting time by dividing the total waiting time by total number of process.

Step 10 : Calculate Average turnaround time by dividing the total time by the number of process.

Step 11 : Display the result.

Program

```
#include<stdio.h>
struct process{
	int id,wait,ser,tottime;
}p[20];

int main(){

	int i,n,j,totalwait=0,totalser=0,avwait;
	printf("enter number of process");
	scanf("%d",&n);

	for(i=1;i<=n;i++){
		printf("enter process_id");
		scanf("%d",&p[i].id);
		printf("enter process service time");
		 scanf("%d",&p[i].ser);
	}
	p[1].wait=0;
	p[1].tottime=p[1].ser;

	for(i=2;i<=n;i++){
		for(j=1;j<i;j++){
			p[i].wait=p[i].wait+p[j].ser;
	}

	totalwait=totalwait+p[i].wait;
	p[i].tottime=p[i].wait+p[i].ser;
	totalser=totalser+p[i].tottime;
}

	avwait=totalwait/n;
	printf("Id\tservice\twait\ttotal");

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

	printf("\n%d\t%d\t%d\t%d\n",p[i].id,p[i].ser,p[i].wait,p[i].tottime);
}

printf("average waiting time %d\n",avwait);

return 0;
}
```

OUTPUT

Shortest Job First

CPU scheduler will decide which process should be given to the CPU for its execution. For this it uses different algorithms to choose among the processes. One among that algorithm is Shortest Job First. In this algorithm the process which has less service time given the CPU after finishing its request only it will allow CPU to execute next other process.

Task 3 : Calculate the Average Time using SJF Algorithm.

Process	Duration	Order	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0

Task 4 Write the output a program for Shortest Job First.

Step 1 : Get the number of process.

Step 2 : Get the id and service time for each process.

Step 3 : Initially the waiting time of first short process as 0 and total time of first short is process the service time of that process.

Step 4 : Calculate the total time and waiting time of remaining process.

Step 5 : Waiting time of one process is the total time of the previous process.

Step 6 : Total time of process is calculated by adding the waiting time and service time of each process.

Step 7 : Total waiting time calculated by adding the waiting time of each process.

Step 8 : Total turnaround time calculated by adding all total time of each process.

Step 9 : Calculate average waiting time by dividing the total waiting time by total number of process.

Step 10 : Calculate average turnaround time by dividing the total waiting time by total number of process.

Step 11 : Display the result.

Program

```
#include<stdio.h>
struct ff{
	int pid,ser,wait;
}p[20];

struct ff tmp;
int main(){
	int i,n,j,tot=0,avwait,totwait=0,tturn=0,aturn;
	printf("Enter the number of process");
	scanf("%d",&n);
	for(i=0;i<n;i++){
		printf("Enter process id");
		scanf("%d",&p[i]);
		printf("Enter service time");
		scanf("%d",&p[i].ser);
		p[i].wait=0;
}

	for(i=0;i<n-1;i++){
		for(j=i+1;j<n;j++){
			if(p[i].ser>p[j].ser){
				tmp=p[i];
				p[i]=p[j];
				p[j]=tmp;
			}
		}
	}

	printf("PID\tSER\tWAIT\tTOT\n");

	for(i=0;i<n;i++){
		tot=tot+p[i].ser;
		p[i+1].wait=tot;
```

```
                totwait=totwait+p[i].wait;

        printf("%d\t%d\t%d\t%d\n",p[i].pid,p[i].ser,p[i].wait,tot);
    }
        avwait=totwait/n;

        printf("TOTAL WAITING TIME:%d\n",totwait);
        printf("AVERAGE WAITING TIME: %d\n",avwait);

    return 0;
    }
```

OUTPUT