B.V.RAJU INSTITUTE OF TECHNOLOGY

VISHNUPUR, NARSAPUR, MEDAK DIST.

(AUTONOMOUS)

STUDENT PROFILE

NAME	:
ROLL NO	:
YEAR	:
B.TECH	:
BATCH	:
LAB NAME	:
ACADEMIC YEAR	:

Signature of Staff - I:

Signature of Staff- II: Signature of HOD

B.V. RAJU INSTITUTE OF TECHNOLOGY

VISHNUPUR, NARSAPUR, MEDAK DIST.

www.bvrit.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Mr. /Miss/Mrs. ……………………………………………………. bearing Regd. No. ………………… of B.Tech ……………… Year ……….. Semester has satisfactorily completed the term work in the **LINUX PROGRAMMING LAB** in the year of 2021-22.

Internal Examiner **External Examiner**

Head of Department

B.V. Raju Institute of Technology

Vishnupur(V), Narsapur(M), Medak (Dist)
(AUTONOMOUS)

INDEX

SNO	NAME OF THE PROGRAM/EXPERIMENT	DATE	PAGE NO	Signature
1	implement basic commands	08/03/22		
2	implerment basic commands for working with directory	15/03/22		
3	a) Write a shell script that accept a file name starting and ending line numbers as arguments and display all the lines between given line no b) Write a shell script that deletes all lines containing a specified word	22/03/22		
4	a) Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions. b) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. whenever the argument is a file, it reports no of lines present in it	29/03/22		
5	a) Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files. b) Write a shell script to list all of the directory files in a director	05/04/22		
6	a) Write a shell script to find factorial of a given number. b) Write an awk script to count number of lines in a file that does not contain vowels	12/04/22		
7	Write an awk script to find the no of characters, words and lines in a file Write a C program to simulate cpu scheduling using first in first out	19/04/22		
8	a) Write a C program that takes one or more file/directory names as command line input and reports the following information A)File Type B)Number Of Links C) Time of last Access D) Read, write, and execute permissions b) Write a C program to list every file in a directory, its inode number, and file name write C program to simulate cpu scheduling using short job first without premption	26/04/22		

9	a) Write a C program to create a child process and allow the parent process to display “parent” and the child to display “child” on the screen b) Write a C program to create a zombie process write C program to simulate cpu scheduling using short job first with premption	24/05/22		
10	Write a C program to illustrate how an orphan process is create Write a C program to simulate cpu scheduling using round robin	31/05/22		
11	Write a C program that illustrate the suspending and resuming process using signal Write a C program to simulate deadlock avoidance using banker’s alogorithm	07/06/22		
12	Write a C program to simulate replacement algorithms using first in first out	14/06/22		
13	Write a C program to simulate replacement algorithms using optimal	14/06/22		
14	Write a C program to simulate replacement algorithms using least recently used	21/06/22		
15	Write a C program to simulate file allocation strategies using squential	21/06/22		
16	Write a C program to simulate file allocation strategies using indexed	28/06/22		
17	Write a C program to simulate file allocation strategies using linked	28/06/22		

WEEK 1: Implement basic commands ***who***, ***date***, ***echo***, ***ls***, ***cat***, ***wc***, ***mv***, ***rm***

Week-1: implement basic commands for working with directory ***Home***, ***pwd***, ***cd***, ***mkdir***, ***cp***, ***subdirectories***, ***ln***.

1) **who**: The who command is used to get information about currently loggedin users onto the system. Syntax: $who [options] [filename]

2) **date**: The date command is used to display and set the date and time on the system on which the operator is operating Linux. To change the date and time super – user (root) must be used.
   Syntax: $date / $date -u / $date - -date = “string”

3) **echo:** The command echo is used in a scripting language and batch files to display a line of text/string on standard output or a file.
   Syntax: echo [string]

4) **ls:** The ls command is used to display a list of content of a directory.
   Syntax: ls

5) **cat:** The cat command is a multi-purpose utility in the Linux system. It can be used to create a file, display the content of the file, copy the content of one file to another file, and more. Syntax: cat > <filename>
   cat <filename>

6) **wc:** The wc command is used to count the lines, words, and characters in a file.
   Syntax: wc<filename>

7) **mv:** The mv command is used to move a file or a directory from one location to another location.
   Syntax: mv <filename> <directory path>

8) **rm**: The rm command is used to remove a file. Syntax: rm <filename>

9) **Home:** The command long with cd will change the directory to home.

   Syntax: cd/home (or) cd $Home

10) **pwd:** The command pwd is used to display the current working directory.

   Syntax: pwd

11) **cd:** The cd command is used to change the current directory.

   Syntax: cd <directory name>

12) **mkdir:** The mkdir command is used to create a new directory.

   Syntax: mkdir <directory name>

13) **cp:** This command is used to copy files or groups of files or directories. It creates an exact image of a file on a disk with a different file name. cp command requires at least two filenames in its arguments.

   Syntax: cp <existing file name> <new file name>

14) **ln:** The ln command is used to create links to files or directories.

   Syntax: ln [file_name] [link_name] {this creates a hard link, the filename is the file for which you want to create a link and link_name is of your wish.

Eg: ln abc.txt abc_link.txt

WEEK 3:

a) Write a shell script that accept a file name starting and ending line numbers as arguments and display all the lines between given line no

b) Write a shell script that deletes all lines containing a specified word a)

```
echo " Enter the file name” read fname echo “enter starting line number”
read sl echo “enter ending line number” read el d=`expr $el - $sl` if [ -f
$fname ] then
echo "the lines between $sl and $el of given file are" head -$el $fname | tail -$ d
else echo "file doesn’t exist"
fi
```

```
INPUT: sh prog1.sh enter
the file name file1 enter
starting line number 15
enter ending line number 20
```

```
OUTPUT    :
# It displays 15 to 20 between lines
```

```
b) if [ $# -ne
0 ] then
echo enter the word read word
for  fname  in  $*  do  if  [  -f
$fname ] then
```

```
echo the given input filename is: $fname grep -v "$word" $fname else

echo its not a file fi done

else

echo "enter at least one argument as input" fi
```

INPUT:

```
sh prog2.sh 3.sh enter

the word echo
```

OUTPUT:

```
he given input filename is: 3.sh
```

It displays all the lines other than pattern matching

WEEK 4:

a) Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions.
b) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or directory and reports accordingly. whenever the argument is a file, it reports no of lines present in it

a)

```
echo "List of Files which have Read, Write and Execute Permissions in Current Directory" for file in * do

if [ -r $file -a -w $file -a -x $file] then

echo $file fi done
```

INPUT: sh

prog3.sh

OUTPUT:

List of Files which have Read, write and Execute Permissions in Current Directory

pp2.txt

b) echo enter the

name for fname in *

do if test -f $fname

then echo "file"

$fname

echo "number of lines" `cat $fname | wc -l` else if test -d $fname

then echo "dir" $fname fi fi done

INPUT:

sh prog4.sh

OUTPUT: enter the name file 3.sh number

of lines 9

WEEK 5

a) Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.
b) Write a shell script to list all of the directory files in a directory a) echo

Enter file name: read file1 read file2

a=`comm -2 $file1 $file2` b=`grep -c $a $file2`

echo Words contained in file one occurred in file two $b times grep -n $a $file2

INPUT: sh prog5.sh

Enter file name: f1

myfile

OUTPUT:

Words contained in file one occurred in file two 3 times 1: myfile contains

5: myfile

8: myfile

b) echo “Enter dir name “

read dir if [-d $dir] then

echo “Files in $dir are “ls

:

```
$dir else echo “Dir does not

exist”

fi
```

INPUT: sh Lp6.sh

Enter dir name Prasanna

OUTPUT:

Files in Prasanna are 3.sh

4.sh pp2.txt

WEEK 6

a) Write a shell script to find factorial of a given number.
b) Write an awk script to count number of lines in a file that does not contain vowels

a) echo Factorial echo

```
Enter number: read n

fact=1 i=1

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

fact=`expr $fact \* $i` done echo

Factorial of $n is $fact
```

INPUT: sh p7.sh Factorial

Enter number:5

OUTPUT:

Factorial of 5 is 120

b)

```
BEGIN {print Displaying number of lines in a file that do not contain vowels
total=0}

{if($0!~/[aeiouAEIOU]/)

total=total + 1}

END{print "The total lines in a file that do not contain v
```

INPUT:

```
awk prog8.awk lp1.sh

Displaying number of lines in a file that do not contain v
```

OUTPUT:

```
The total lines in a file that do not contain vowels:1
```

:

WEEK 7

Write an awk script to find the no of characters, words and lines in a file

```
BEGIN{ print Displaying number of characters, words and lines in a fi e}
{word=words + NF}

{len = length($0)} {charcount=charcount + len}

END{print The total number of characters, words and lines in a file is:
print("Words:\t", words)

print("Lines:\t", NR) print("Chars:\t",len) }
```

INPUT:

```
awk prog9.awk lp5.sh
```

OUTPUT:

```
The total number of characters, words and lines in a file is: Words:12

Lines:3

Chars:39
```

WEEK 8:

a) Write a C program that takes one or more file/directory names as command line input and reports the following information A)File Type B)Number Of Links C) Time of last Access D) Read, write, and execute permissions
b) Write a C program to list every file in a directory, its inode number, and file name

a)

```
#include<stdio.h>

int main()

{

FILE *stream; int buffer_character; stream=fopen("test","r");

if(stream==(FILE*)0)

{ printf(stderr,"Error opening file(printed to standard error)\n");

fclose(stream); exit(1); }}

if(fclose(stream))==EOF)

{ printf(stderr,"Error closing stream.(printed to standard error)\n);

exit(1);
```

```
        :
}
return();
}
```

b)

```
#include<stddef.h>

#include<stdio.h>

#include<sys/types.h

> #include<dirent.h>

int main()

{

DIR *dp; Struct dirent *p; char dname[20];

Struct stat x;

Printf("Enter the directory name:");

Scanf("%s",dname);

Dp=opendir(dname);

Printf("\n FILE NAME \t INODE NUMBER \n");
While((p=readdir(dp))!=NULL)

{

Printf("%s\t %/d\n",p->d name,x.stino);

}

}
```

INPUT:

```
cc inode.c –o inode

./inode
```

OUTPUT :

```
FILE NAME    INODE NUMBER
…………..     4195164
File2.c    4195164
….
File1.c    4195164
```

WEEK 9:

a) Write a C program to create a child process and allow the parent process to display “parent” and the child to display “child” on the screen
b) Write a C program to create a zombie process a)

```
#include <stdio.h>

int   main()   {   int

pid;

printf("I'm the original process with PID %d and PPID
%d.\n",getpid(),getppid());

pid=fork(); /* Duplicate. Child and parent continue from here.*/ if

(pid!=0)  {/* pid is non-zero, so I must be the parent  */

printf("I'm the parent process with PID %d and PPID

%d.\n",getpid(),getppid()); printf("My child's PID is %d.\n", pid);

}
else { /* pid is zero, so I must be the child. */
printf("I'm the child process with PID %d and PPID %d.\n",getpid(),getppid());
}
printf("PID %d terminates.\n",pid); /* Both processes execute this */ }
```

INPUT:

```
$cc fork.c          ... run the program.
./a.out
```

OUTPUT :

```
I'm the original process with PID 13292 and PPID 13273.
I'm the parent process with PID 13292 and PPID 13273.
My child's PID is 13293.
I'm the child process with PID 13293 and PPID 13292.
PID 13293 terminates.          ... child terminates.
PID 13292 terminates.          ... parent terminates.
```

b)

```
#include <stdio.h>
int main(){ int pid;
pid=fork(); /* Duplicate */ if (pid!=0) /* Branch based on
return value from fork() */    { while (1) /* never terminate,
and never execute a wait() */ sleep(1000);          }
else         {
exit(42); /* Exit with a silly number */
}}
```

INPUT:

```
$ cc prog17.c
./a.out& ... execute the program in the background.
JNTU[1]13545
```

OUTPUT:

```
 $ ps
PID TT STAT TIME COMMAND
13535 p2 s         0:00 -ksh(ksh) ... the shell
13545 p2 s        0:00 aombie.exe... the parent process
13536 p2 z        0:00 <defunct> ... the zombie child process 13537 p2 R    0:00
ps
$ kill 13545      ... kill the parent process.
[1]   erminated  zombie.exe
$ ps         ... notice the zombie is gone now.
     PID TT STAT TIME COMMAND
     13535 p2 s      0:00 -csh(csh)
     13548 p2 R      0:00 ps
```

WEEK 10: Write a C program to illustrate how an orphan process is created

```
 #include <stdio.h>
int main() { int
pid;
printf("I'm the original process with PID %d and PPID %d.\n", getpid(),getppid());
pid=fork(); /* Duplicate. Child and parent continue fr  m here.*/ if (pid!=0) /*
Branch based on return value fr           m f rk() */ { /* pid is non-zero, so I must be
the parent    */      printf("I'm the parent process with PID %d and PPID %d.\n",
getpid(),getppid());
        printf("My child's PID is %d.\n", pid);
}
else    { /* pid is zero, so I must be the child. */
sleep(5); /*Make sure that the parent terminates first. */
printf("I'm the child process with PID %d and PPID %d.\n", getpid(),getppid());
}
printf("PID %d terminates.\n",pid); /* Both processes execute this */}
```

INPUT :

```
$cc prog18.c
./a.out ... run the program.
```

OUTPUT :

```
I'm the original process with PID 13364 and PPID 13346.
I'm the parent process with PID 13364 and PPID 13346.
PID 13364 terminates.
I'm the child process with PID 13365 and PPID 1. ...orphaned!
PID 13365 terminates.     ... child terminates.
```

WEEK 12: Write a C program that illustrate the suspending and resuming process using signal

```
#include<sys/types.h>
#include<signal.h>
//suspend the process(same as hitting crtl+z) kill(pid,SIGSTOP);
//continue the process kill(pid,SIGCONT);
```

Aim: Simulate the following CPU Scheduling Algorithm

a. FCFS(First come first serve)

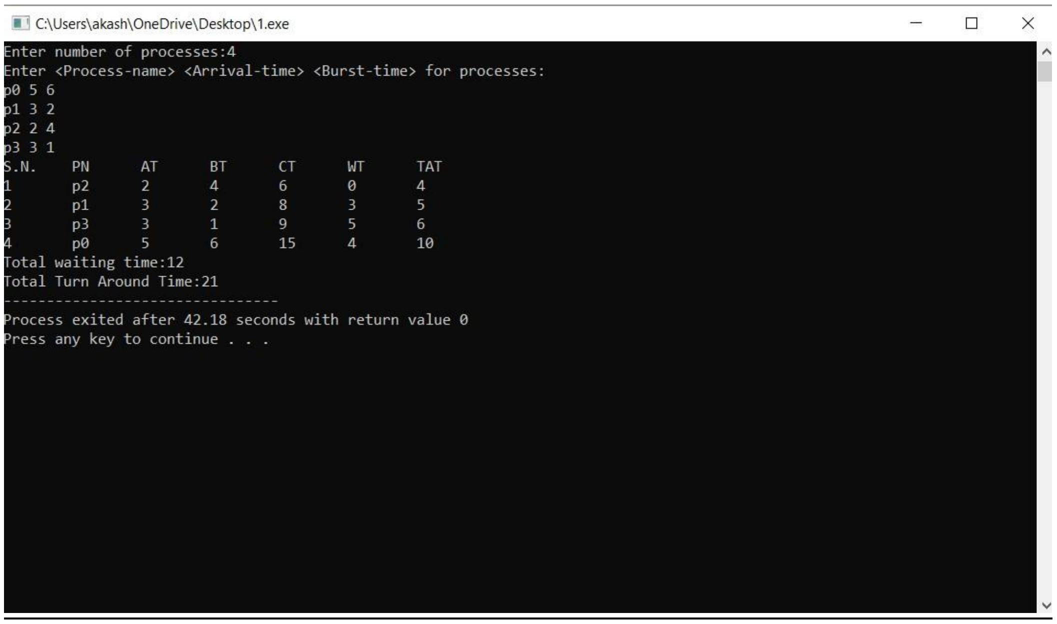
```
//Program For “First-Come-First-Serve "CPU Scheduling Algorithm

#include<stdio.h>

#include<string.h> int

main(void)

{

//VARIABLE DECLARATION char pn[20][20], c[20][20]; //PN-PROGRAM NAMES

int n,i,j,at[20], bt[20], wt[20],tat[20], ct[20]; //bt-Burst Time ; wt-Waiting Time;

// tat-Turn Around Time int twt=0,

ttat=0, temp1, temp2; printf("Enter

number of processes:"); scanf("%d",

&n);

//taking input values i.e., process-names, arrival-times and burst-times

printf("Enter <Process-name> <Arrival-time> <Burst-time> for processes:\n", (i+1));

for(i=0; i<n; i++) scanf("%s%d%d",&pn[i],&at[i],&bt[i]); //Sort The Processes

According To Arrival Times for(i=0;i<n;i++)

{

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

{ if(at[i]>at[j])

{

temp1 = bt[i];

temp2 = at[i];

bt[i] = bt[j]; at[i] =

at[j]; bt[j] =

temp1; at[j] =
temp2;

strcpy(c[i],pn[i]);

strcpy(pn[i],pn[j]);

strcpy(pn[j],c[i]);

}

}

if(i==0) ct[0]=at[0]+bt[0];

if(i>0) { if(at[i]>ct[i-1])

ct[i]=at[i]+bt[i]; else

ct[i]=ct[i-1]+bt[i];

}

}

//Calculating Waiting Time & Tat

wt[0]=0; tat[0]=ct[0]-at[0];

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

{ tat[i] = ct[i]-

at[i]; wt[i] =

tat[i]-bt[i]; twt +=

wt[i]; ttat +=

tat[i];

}

//Printing The Values After All Preocesses Completed

printf("S.N.\tPN\tAT\tBT\tCT\tWT\tTAT\n"); for(i=0; i<n; i++)

printf("%d\t%s\t%d\t%d\t%d\t%d\t%d\n",(i+1),pn[i],at[i],bt[i],ct[i],wt[i],tat[i])
```

; printf("Total waiting time:%d\n", twt); printf("Total Turn Around Time:%d",

ttat);

}

OUTPUT



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter number of processes:4
Enter <Process-name> <Arrival-time> <Burst-time> for processes:
p0 5 6
p1 3 2
p2 2 4
p3 3 1
S.N.    PN      AT      BT      CT      WT      TAT
1       p2      2       4       6       0       4
2       p1      3       2       8       3       5
3       p3      3       1       9       5       6
4       p0      5       6       15      4       10
Total waiting time:12
Total Turn Around Time:21
--------------------------------
Process exited after 42.18 seconds with return value 0
Press any key to continue . . .
```

b. **SJF(Shortest job first)(Without preemption)**

Program

//PROGRAM FOR SHORTEST-JOB-FIRST(SJF) "CPU SCHEDULING ALGORITHM" WITHOUT PRE_EMPTION

#include<stdio.h> //

#include<conio.h>

int main()

{

int at[10], bt[10], ct[10], wt[10], ta[10], tat[10];

//at-ArritvalTime::br-BurstTime::ct-CompletionTime::ta-TemporaryArray

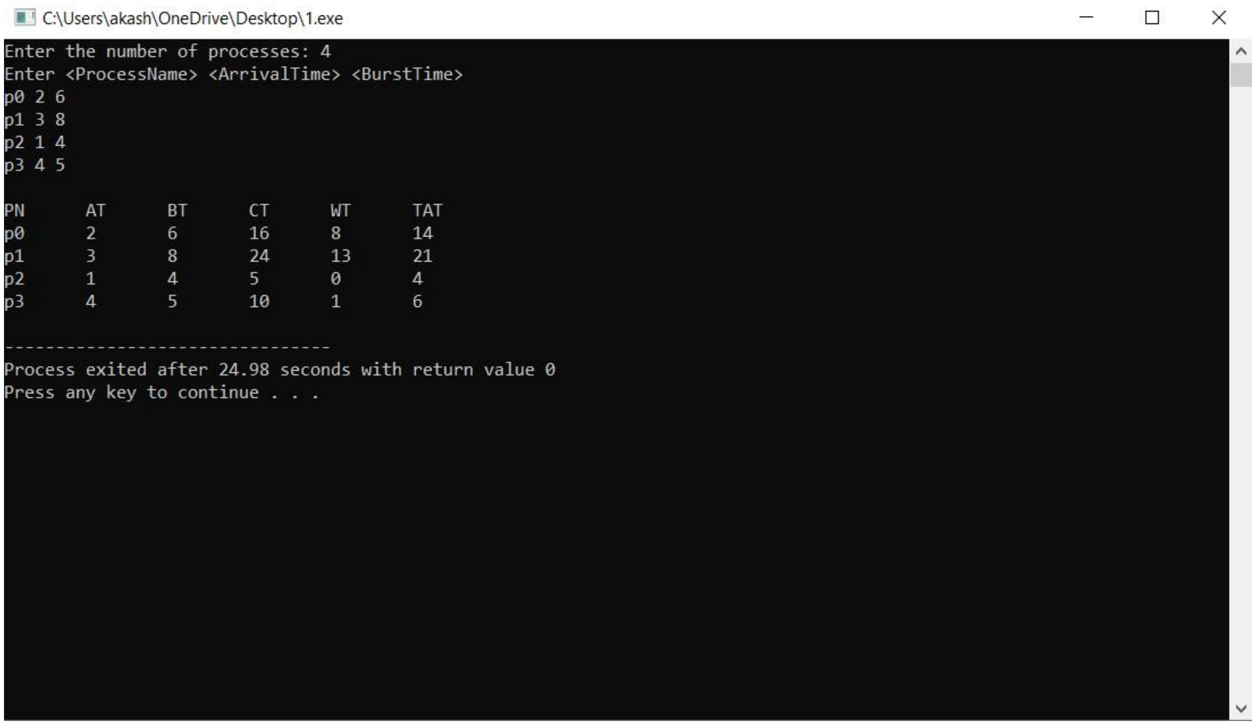
//wt-WaitingTime::tat-TurnAroundTime::tn-CurrentTime(TimeNow) int

n, i, k, pc=0, pointer = 0, tn =0, c;//pc-ProcessesCompleted char

```
pn[10][10]; //pn-ProcessName printf("Enter the number of processes:

"); scanf("%d",&n); printf("Enter <ProcessName> <ArrivalTime>

<BurstTime>\n"); for(i=0;i<n;i++) scanf("%s%d%d",&pn[i],&at[i],&bt[i]);

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

{ ct[i] = -1;

ta[i] = bt[i];

}

while(pc!=n)
{ c = 0; for(i=0; i<n;

i++) if(ct[i]<0 &&

at[i]<=tn) c++; if(c==0)

tn++; else

{

pointer = 0; while(at[pointer]>tn ||

ct[pointer]>0) pointer++; for(k=pointer+1;

k<n; k++) if(at[k]<=tn && ct[k]<0 &&

bt[pointer]>bt[k]) pointer = k;

if(ct[pointer]<0)

{

tn=tn+bt[pointer]; bt[pointer] = 0; ct[pointer] = tn;

wt[pointer] = ct[pointer] - ( at[pointer]+ ta[pointer] );

tat[pointer] = ct[pointer] - at[pointer]; pc++;

}
}

}
```

```
printf("\nPN\tAT\tBT\tCT\tWT\tTAT\n"); for(i=0;i<n;i++)

printf("%s\t%d\t%d\t%d\t%d\t%d\n",pn[i],at[i],ta[i],ct[i],wt[i],tat[i])

; return 0;}
```

OUTPUT



```
Enter the number of processes: 4
Enter <ProcessName> <ArrivalTime> <BurstTime>
p0 2 6
p1 3 8
p2 1 4
p3 4 5

PN      AT      BT      CT      WT      TAT
p0      2       6       16      8       14
p1      3       8       24      13      21
p2      1       4       5       0       4
p3      4       5       10      1       6

--------------------------------
Process exited after 24.98 seconds with return value 0
Press any key to continue . . .
```

c. SJF(Shortest job first)(With preemption)

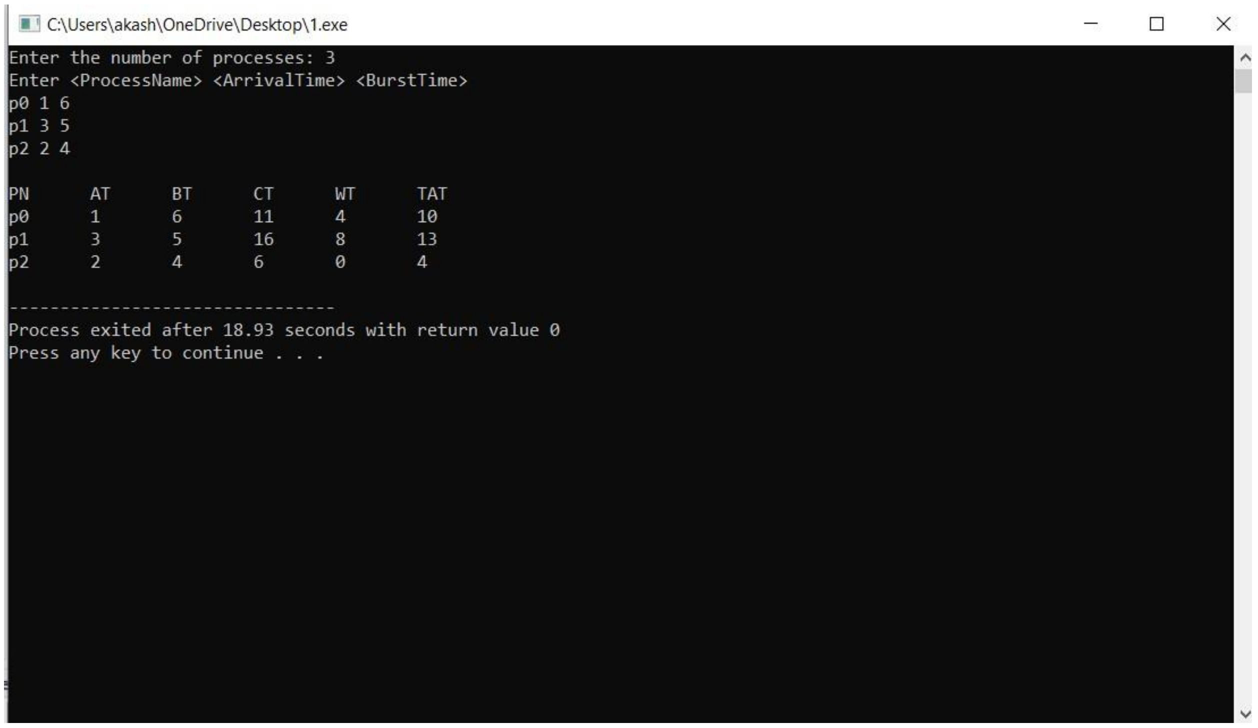
Program

```
// PROGRAM FOR SHORTEST-JOB-FIRST(SJF) "CPU SCHEDULING ALGORITHM" WITH
PRE_EMPTION

#include<stdio.h> //

#include<conio.h>

int main()

{

int n,at[10], bt[10], ct[10], wt[10], tn =0, c, ta[10],tat[10];

//at-ArritvalTime::br-BurstTime::ct-CompletionTime::ta-TemporaryArray

//wt-WaitingTime::tat-TurnAroundTime::tn-CurrentTime(TimeNow)

// int i, j, k, tot, pc=0, pointer = 0, lp=-1;//lp-Last-executedProcess

int i, k, tot, pc=0, pointer = 0, lp=-1;//lp-Last-executedProcess

char pn[10][10]; printf("Enter the number of processes: ");

scanf("%d",&n); printf("Enter <ProcessName> <ArrivalTime>

<BurstTime>\n"); for(i=0;i<n;i++)

scanf("%s%d%d",&pn[i],&at[i],&bt[i]); for(i=0; i<n; i++)

{ ct[i] = -1;

ta[i] = bt[i];

}

while(pc!=n)

{

k=0; c = 0;

for(i=0; i<n;

i++)

{
```

```
if(ct[i]<0 && at[i]<=tn) c++;

}

if(c==0)

tn++; else

{

pointer = 0; while(at[pointer]>tn ||

ct[pointer]>0) pointer++;

for(k=pointer+1; k<n; k++) if(

(at[k]<=tn && ct[k]<0) &&

( (bt[pointer]==bt[k] && k==lp) || bt[pointer]>bt[k] ) )

pointer = k; if(ct[pointer]<0)

{

bt[pointer]--;
tn++; if(bt[pointer]==0)

{

ct[pointer] = tn; wt[pointer] = ct[pointer] - (

at[pointer]+ ta[pointer] ); tat[pointer] = ct[pointer] -

at[pointer]; pc++;

}

}

lp = pointer;

}

}
```

```
printf("\nPN\tAT\tBT\tCT\tWT\tTAT\n"); for(i=0;i<n;i++)

printf("%s\t%d\t%d\t%d\t%d\t%d\n",pn[i],at[i],ta[i],ct[i],wt[i],tat[i])

; return 0;

}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter the number of processes: 3
Enter <ProcessName> <ArrivalTime> <BurstTime>
p0 1 6
p1 3 5
p2 2 4

PN      AT      BT      CT      WT      TAT
p0      1       6       11      4       10
p1      3       5       16      8       13
p2      2       4       6       0       4

--------------------------------
Process exited after 18.93 seconds with return value 0
Press any key to continue . . .
```

d. Round Robin

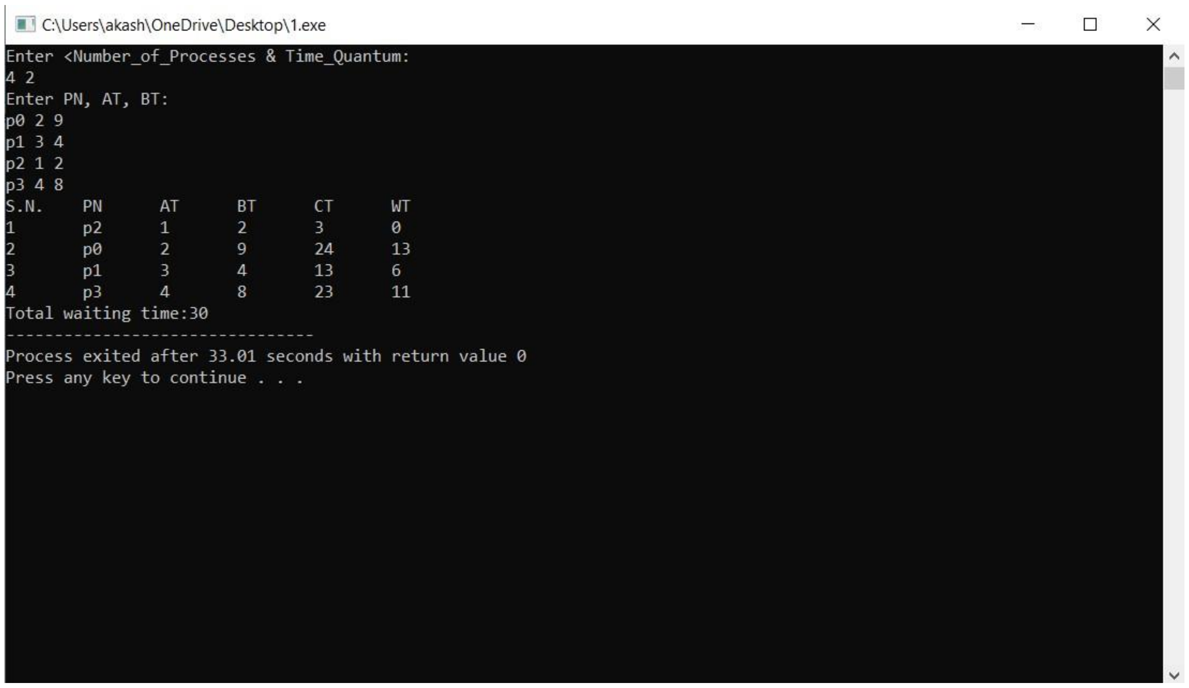
Program

```
//PROGRAM FOR ROUND ROBIN "CPU SCHEDULING ALGORITHM" WITH ARRIVAL TIMES

#include<stdio.h>

#include<string.h> int

main(void)

{

//VARIABLE DECLARATION char pn[20][20],

c[20][20]; //PN-PROGRAM NAMES

int n,i,j,k,l, tq, at[20], bt[20], rbt[20], wt[20],tt[20],ct[20]; //bt-BURST TIME ; wt-WAITING TIME;

tat-TURN AROUND TIME int temp1, temp2, temp3, count=0,twt=0, tn;//,tat=0; printf("Enter

<Number_of_Processes & Time_Quantum:\n"); scanf("%d%d", &n, &tq); printf("Enter PN, AT,

BT:\n");

//TAKING INPUT VALUES i.e., PROCESS-NAMES, ARRIVAL-TIMES, BURST-TIMES

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

scanf("%s%d%d",&pn[i],&at[i],&bt[i]);

for(i=0; i<n; i++) rbt[i]=bt[i];

//SCHEDULING THE PROCESSES ACCORDING TO SJF

for(i=0;i<n;i++)
{ for(j=i+1;

j<n;j++)

{

//SORTING BASED ON ARRIVAL TIMES
```

```
if(at[i]>at[j])

{

temp1 = bt[i];

bt[i]   =   bt[j];

bt[j] = temp1;

temp2 = at[i];

at[i] = at[j]; at[j] =

temp2; temp3 =

rbt[i]; rbt[i] =

rbt[j]; rbt[j] =

temp3;

strcpy(c[i],pn[i]);

strcpy(pn[i],pn[j]);

strcpy(pn[j],c[i]);

}

} //END OF J FOR-LOOP }//END

OF I FOR-LOOP

tn = at[0]; label:

for(i=0; i<n; i++)
{ if(at[i]>tn) i-

-; if(rbt[i]>0)

{ if(rbt[i]>tq)

{
```

```
tn += tq; rbt[i]

-= tq;

}

else

{

tn += rbt[i];

rbt[i]   =   0;

ct[i] = tn;

count++;

}

}

}

if(count<n) goto label;

//CALCULATING WAITING TIME & TAT

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

{ wt[i] = ct[i]-at[i]-

bt[i]; twt += wt[i];

}

//PRINTING THE VALUES AFTER ALL PREOCESSES COMPLETED

printf("S.N.\tPN\tAT\tBT\tCT\tWT\n"); for(i=0; i<n; i++)

printf("%d\t%s\t%d\t%d\t%d\t%d\n",(i+1),pn[i],at[i],bt[i],ct[i],wt[i]);

printf("Total waiting time:%d", twt);

} //END OF MAIN
```

OUTPUT



```
Enter <Number_of_Processes & Time_Quantum:
4 2
Enter PN, AT, BT:
p0 2 9
p1 3 4
p2 1 2
p3 4 8
S.N.    PN      AT      BT      CT      WT
1       p2      1       2       3       0
2       p0      2       9       24      13
3       p1      3       4       13      6
4       p3      4       8       23      11
Total waiting time:30
--------------------------------
Process exited after 33.01 seconds with return value 0
Press any key to continue . . .
```

Aim: Deadlock Avoidance

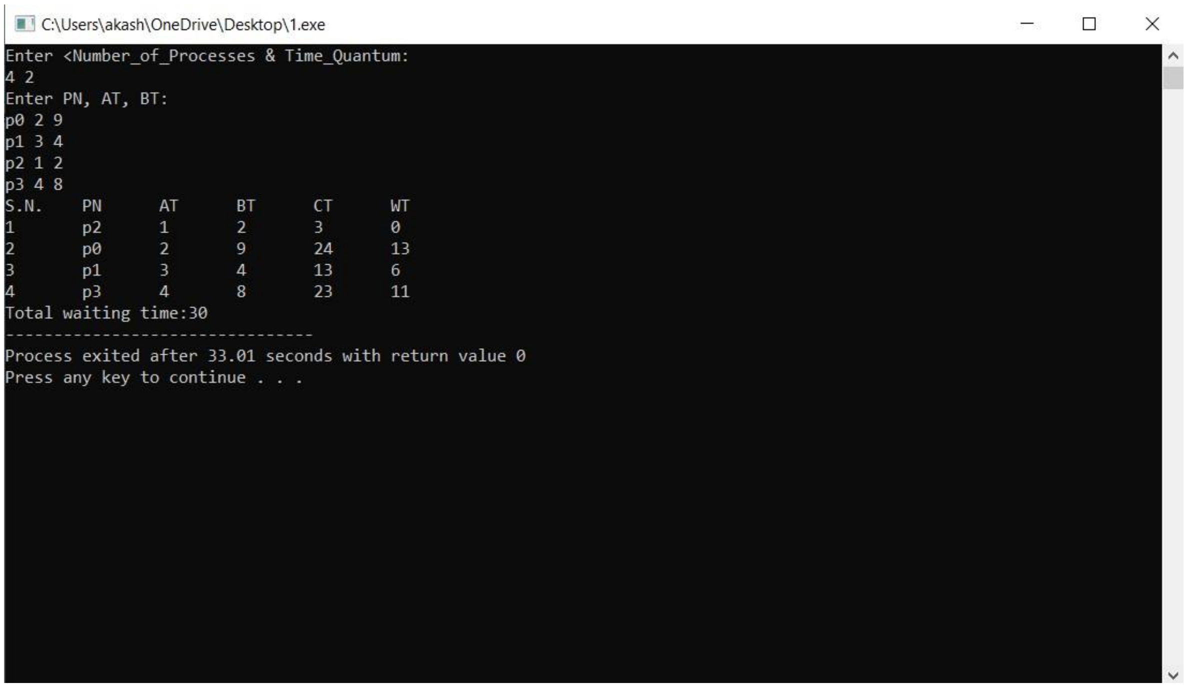
a. Bankers Algorithm

Program

```
// Banker's Algorithm #include
<stdio.h>
int main()
{
// P0, P1, P2, P3, P4 are the Process names here

int n, m, i, j, k;
```

```
n = 5; // Number of processes m
= 3; // Number of resources
int alloc[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix
{ 2, 0, 0 }, // P1
{ 3, 0, 2 }, // P2
{ 2, 1, 1 }, // P3
{ 0, 0, 2 } }; // P4

int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix
{ 3, 2, 2 }, // P1
{ 9, 0, 2 }, // P2
{ 2, 2, 2 }, // P3
{ 4, 3, 3 } }; // P4

int avail[3] = { 3, 3, 2 }; // Available Resources

int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++) { f[k]
= 0;
}
int need[n][m];
for (i = 0; i < n; i++) { for
(j = 0; j < m; j++)
need[i][j] = max[i][j] - alloc[i][j];
} int y = 0; for (k = 0; k
< 5; k++) { for (i = 0; i
< n; i++) {
if (f[i] == 0) {

int flag = 0; for (j = 0; j
< m; j++) { if
(need[i][j] > avail[j]){
flag = 1; break;
}
}

if (flag == 0) {
ans[ind++] = i; for (y
= 0; y < m; y++)
avail[y] += alloc[i][y];
f[i] = 1; }
}
}
}

int flag = 1;

for(int i=0;i<n;i++)
{ if(f[i]==0)
{
flag=0;
printf("The following system is not safe"); break;
}
}

if(flag==1)
{
printf("Following is the SAFE Sequence\n");
for (i = 0; i < n - 1; i++) printf(" P%d ->",
ans[i]);
printf(" P%d", ans[n - 1]);
}
return (0);
}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter <Number_of_Processes & Time_Quantum:
4 2
Enter PN, AT, BT:
p0 2 9
p1 3 4
p2 1 2
p3 4 8
S.N.    PN      AT      BT      CT      WT
1       p2      1       2       3       0
2       p0      2       9       24      13
3       p1      3       4       13      6
4       p3      4       8       23      11
Total waiting time:30
--------------------------------
Process exited after 33.01 seconds with return value 0
Press any key to continue . . .
```

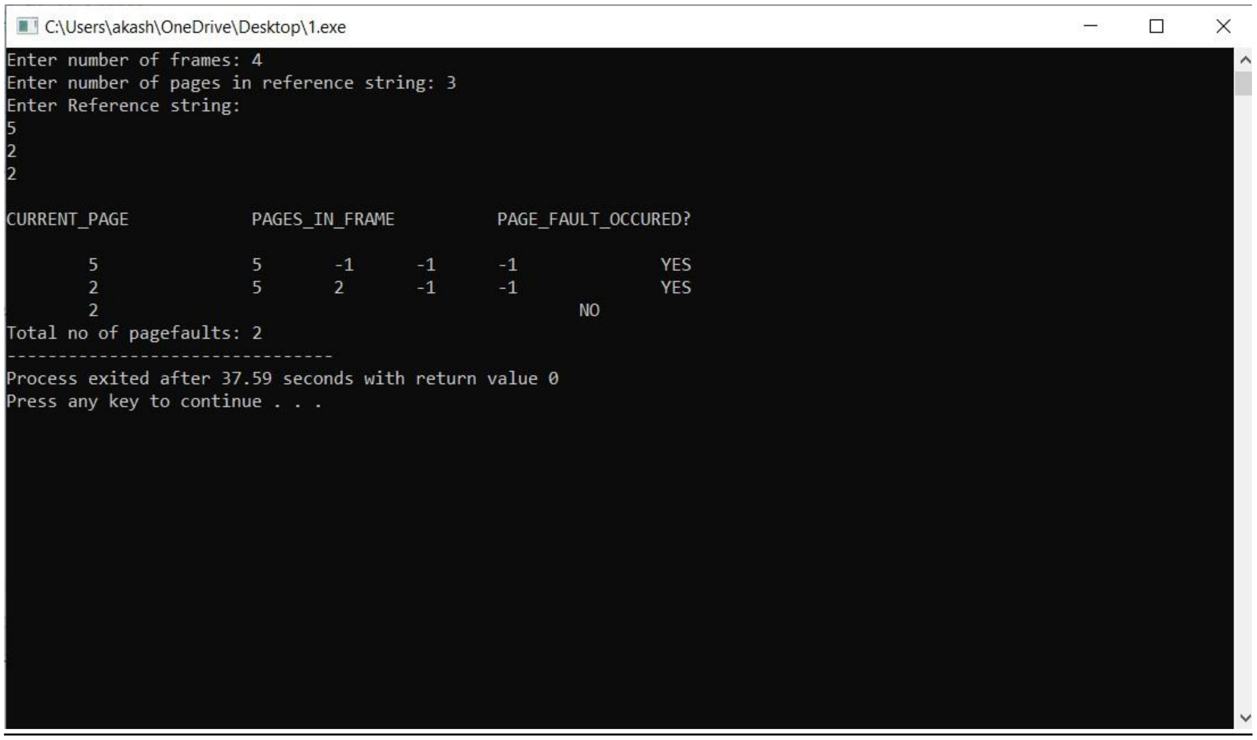
Aim: Replacement Algorithms

a. First in first out

Program

```
#include <stdio.h>

#include <conio.h>  int

main()

{

 int n, rss, fa[20], rsa[50]; //n-No_of_Frames

 //rss->Reference_String_Size::fa->Frame_Array   //rsa-

>Reference_String_Array::ta->Temporary_Array   int

i,j,k,pfc=0,npf, cp=0;

 //cp->Current_Position :: ff->Frames_Filled ::pfc->Page_Fault_Count

 //npf:NO_Page_Faults [0-False, 1-True]

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

printf("Enter number of pages in reference string: ");

scanf("%d", &rss);  printf("Enter Reference

string:\n");  for(i=0; i<rss; i++)  scanf("%d",&rsa[i]);

for(i=0;i<n;i++)  fa[i]=-1;

 printf("\nCURRENT_PAGE\t\tPAGES_IN_FRAME\t\tPAGE_FAULT_OCCURED?\n");

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

 printf("\n\t%d\t\t",rsa[i]);  npf=0; for(j=0;j<n;j++)

//Checking for the page in FRAME ARRAY

 {

 if(fa[j]==rsa[i])
 {
npf = 1;
printf("\t\t\t\tNO"); break;
 }
 }
 if(npf==0) // if page fault occurs
 {
 pfc++;  fa[cp] =
rsa[i];
cp=(cp+1)%n;
for(j=0;j<n;j++)
printf("%d\t",fa[j]);
printf("\tYES");
 }
 }
 printf("\nTotal no of pagefaults: %d",pfc);
return 0;
}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter number of frames: 4
Enter number of pages in reference string: 3
Enter Reference string:
5
2
2

CURRENT_PAGE            PAGES_IN_FRAME          PAGE_FAULT_OCCURED?

        5               5       -1      -1      -1              YES
        2               5       2       -1      -1              YES
        2                                               NO
Total no of pagefaults: 2
--------------------------------
Process exited after 37.59 seconds with return value 0
Press any key to continue . . .
```

b. Optimal

Program

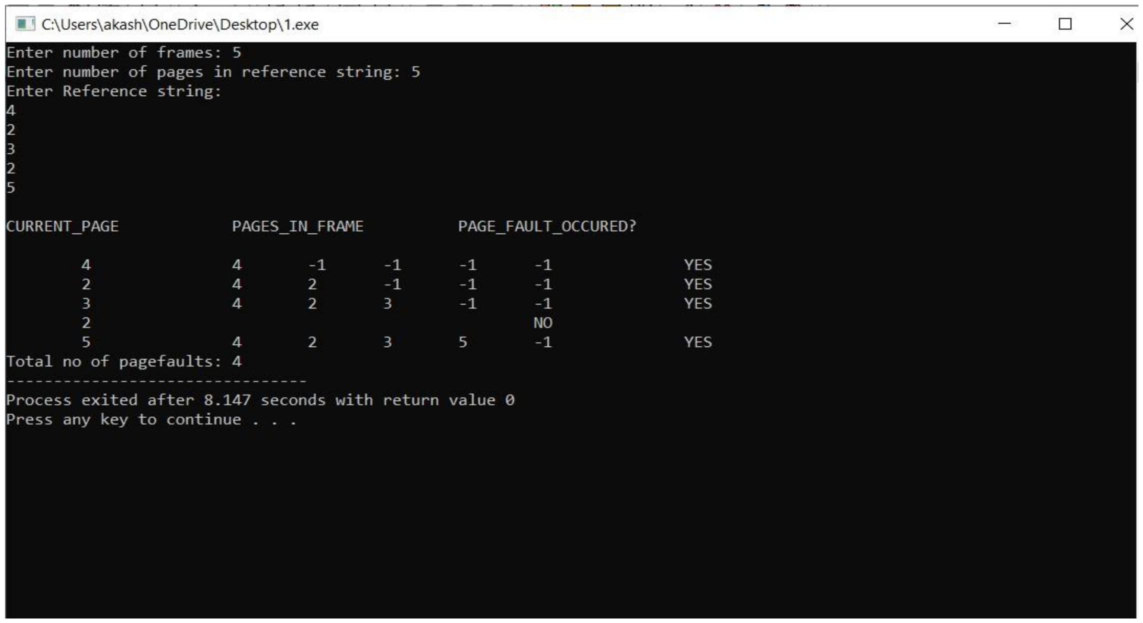
```
include <stdio.h>

#include <conio.h>  int

main()

{

 int n, rss, fa[20], rsa[50], ta[20]; //n-No_of_Frames   //rss->Reference_String_Size::fa-

>Frame_Array   //rsa->Reference_String_Array::ta->Temporary_Array

 int i,j,k, d,pfc=0,npf, cp,ff=0;

 //d-distance[How soon a page will be used again?]

 //cp->Current_Position :: ff->Frames_Filled ::pfc->Page_Fault_Count

 //npf:NO_Page_Faults [0-False, 1-True]

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

printf("Enter number of pages in reference string: ");

scanf("%d", &rss);   printf("Enter Reference

string:\n");   for(i=0; i<rss; i++)   scanf("%d",&rsa[i]);

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

 {

 fa[i]=-1;
ta[i]=999;

}


printf("\nCURRENT_PAGE\t\tPAGES_IN_FRAME\t\tPAGE_FAULT_OCCURED?\n"

);

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

 {
```

```
printf("\n\t%d\t\t",rsa[i]);   npf=0;   for(j=0;j<n;j++)

//Checking for the page in FRAME ARRAY

{

if(fa[j]==rsa[i])

{

npf = 1;

printf("\t\t\t\tNO");   break;

}

}

if(npf==0) // if page fault occurs

{

pfc++;

if(ff<n)

{

fa[ff]=rsa[i];
ff++;

}

else

{

for(k=0;k<n;k++)   ta[k]=999;   for(k=0; k<n;

k++) //finding how near a page is

{

d = 0; // d-> distance

for(j=i+1;j<rss;j++)
```

```
 {
 if(fa[k]==rsa[j])
 {
 ta[k]=d;
break;
 }
 else
d++;
 }
 }
 cp=0;
for(j=1;j<n;j++)
{
if(ta[cp]<ta[j])
cp=j; //cp->current position
}
 fa[cp] = rsa[i];
 }
 for(j=0;j<n;j++)
printf("%d\t",fa[j]);
printf("\tYES");
 }
 }
```

```
printf("\nTotal no of pagefaults: %d",pfc);

return 0;

}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter number of frames: 5
Enter number of pages in reference string: 5
Enter Reference string:
4
2
3
2
5

CURRENT_PAGE            PAGES_IN_FRAME          PAGE_FAULT_OCCURED?

       4               4       -1      -1      -1      -1              YES
       2               4       2       -1      -1      -1              YES
       3               4       2       3       -1      -1              YES
       2                                               NO
       5               4       2       3       5       -1              YES
Total no of pagefaults: 4
--------------------------------
Process exited after 8.147 seconds with return value 0
Press any key to continue . . .
```

C. L R U

PROGRAM

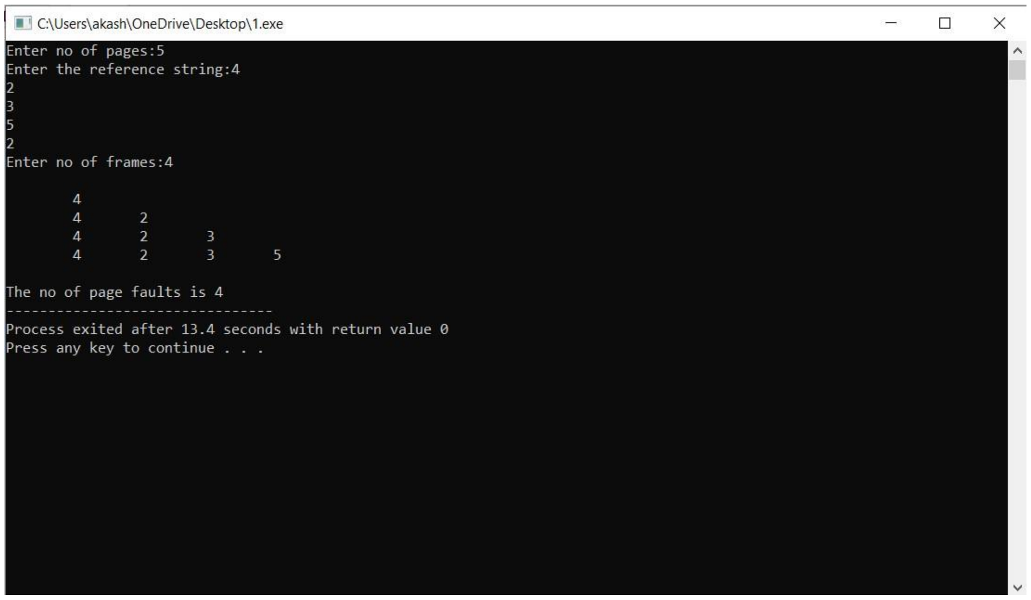
```
#include<stdio.h> main()

{

        int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];

        printf("Enter no of pages:"); scanf("%d",&n);

        printf("Enter the reference string:");

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

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

        printf("Enter no of frames:");
```

```
            scanf("%d",&f); q[k]=p[k];

            printf("\n\t%d\n",q[k]);

            c++; k++;

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

            )

            {

    c1=0;

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

                {

    if(p[i]!=q[j])

    c1++;

                }
                if(c1==f)              {

    c++;                    if(k<f)

    {                         q[k]=p[i];

    k++;

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

    printf("\t%d",q[j]);

    printf("\n");

                    }

    else                    {

    for(r=0;r<f;r++)

                              {

    c2[r]=0;                            for(j=i-

    1;j<n;j--)

                                  {

    if(q[r]!=p[j])
```

```
c2[r]++;
else                                              break;
                                   }
}
for(r=0;r<f;r++)
b[r]=c2[r];
for(r=0;r<f;r++)
                          {
for(j=r;j<f;j++)
                                                            {
     if(b[r]<b[j])                                          {
                                                      t=b[r];
                                                   b[r]=b[j];
                                                       b[j]=t;
                                            }
                                   }
}
for(r=0;r<f;r++)
                          {
if(c2[r]==b[0])
q[r]=p[i];
printf("\t%d",q[r]);
                          }
printf("\n");
                  }
         }
```

```
}
printf("\nThe no of page faults is %d",c);
}
```

OUTPUT :



C:\Users\akash\OneDrive\Desktop\1.exe

```
Enter no of pages:5
Enter the reference string:4
2
3
5
2
Enter no of frames:4

        4
        4       2
        4       2       3
        4       2       3       5

The no of page faults is 4
--------------------------------
Process exited after 13.4 seconds with return value 0
Press any key to continue . . .
```

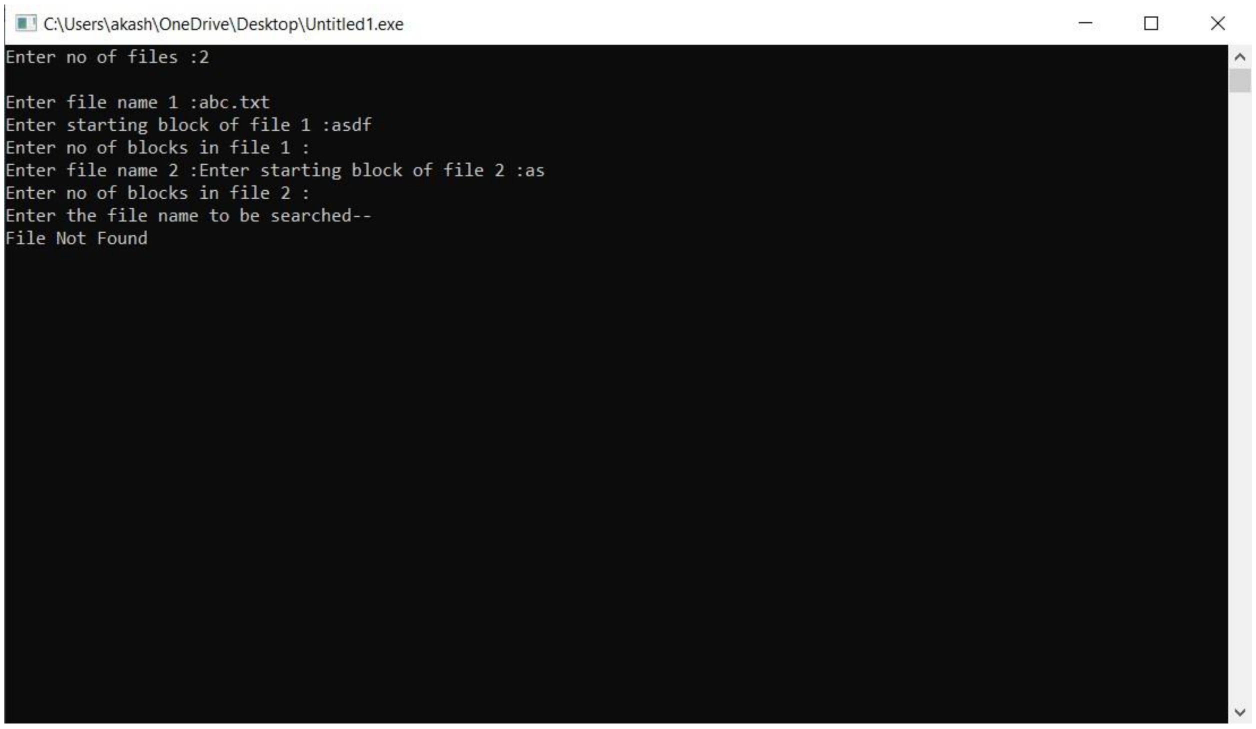
Aim:File Allocation Strategies

a. Sequential

Program

```
#include<stdio.h> #include<conio.h>
struct fileTable
{
char name[20];
int sb, nob;
}ft[30]; void
main()
{ int i, j,
n;
char s[20]; clrscr();
printf("Enter no of files :"); scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter file name %d :",i+1); scanf("%s",ft[i].name);
printf("Enter starting block of file %d :",i+1); scanf("%d",&ft[i].sb);
printf("Enter no of blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
}
printf("\nEnter the file name to be searched-- ");
scanf("%s",s); for(i=0;i<n;i++) if(strcmp(s,
ft[i].name)==0) break; if(i==n)
printf("\nFile Not Found");
else
{
printf("\nFILE NAME START BLOCK NO OF BLOCKS BLOCKS OCCUPIED\n");
printf("\n%s\t\t%d\t\t%d\t",ft[i].name,ft[i].sb,ft[i].nob); for(j=0;j<ft[i].nob;j++) printf("%d,
",ft[i].sb+j);
}
getch();
}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\Untitled1.exe

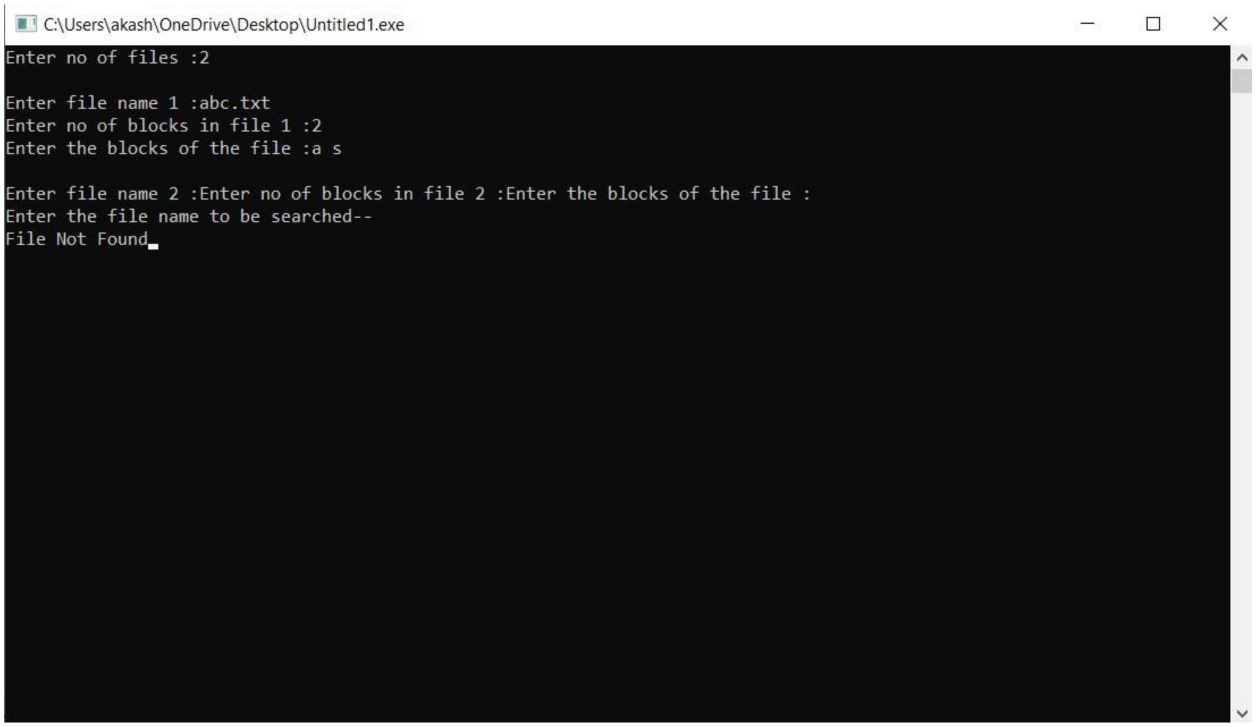
```
Enter no of files :2

Enter file name 1 :abc.txt
Enter starting block of file 1 :asdf
Enter no of blocks in file 1 :
Enter file name 2 :Enter starting block of file 2 :as
Enter no of blocks in file 2 :
Enter the file name to be searched--
File Not Found
```

b. Indexed:

Program

```
#include<stdio.h> #include<conio.h>
struct fileTable
{
char name[20];
int nob, blocks[30];
}
ft[30]; void
main()
{
int i, j, n;
char      s[20];      clrscr();
printf("Enter no of files :");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter file name %d :",i+1);
scanf("%s",ft[i].name); printf("Enter no of
blocks in file %d :",i+1);
scanf("%d",&ft[i].nob); printf("Enter the
blocks of the file :");
for(j=0;j<ft[i].nob;j++)
scanf("%d",&ft[i].blocks[j]);
}
printf("\nEnter the file name to be searched-- ");
scanf("%s",s); for(i=0;i<n;i++) if(strcmp(s,
ft[i].name)==0) break; if(i==n) printf("\nFile Not
Found");
else
{
printf("\nFILE NAME NO OF BLOCKS BLOCKS OCCUPIED"); printf("\n
%s\t\t%d\t",ft[i].name,ft[i].nob); for(j=0;j<ft[i].nob;j++)
printf("%d, ",ft[i].blocks[j]);
}
getch();
}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\Untitled1.exe

```
Enter no of files :2

Enter file name 1 :abc.txt
Enter no of blocks in file 1 :2
Enter the blocks of the file :a s

Enter file name 2 :Enter no of blocks in file 2 :Enter the blocks of the file :
Enter the file name to be searched--
File Not Found
```

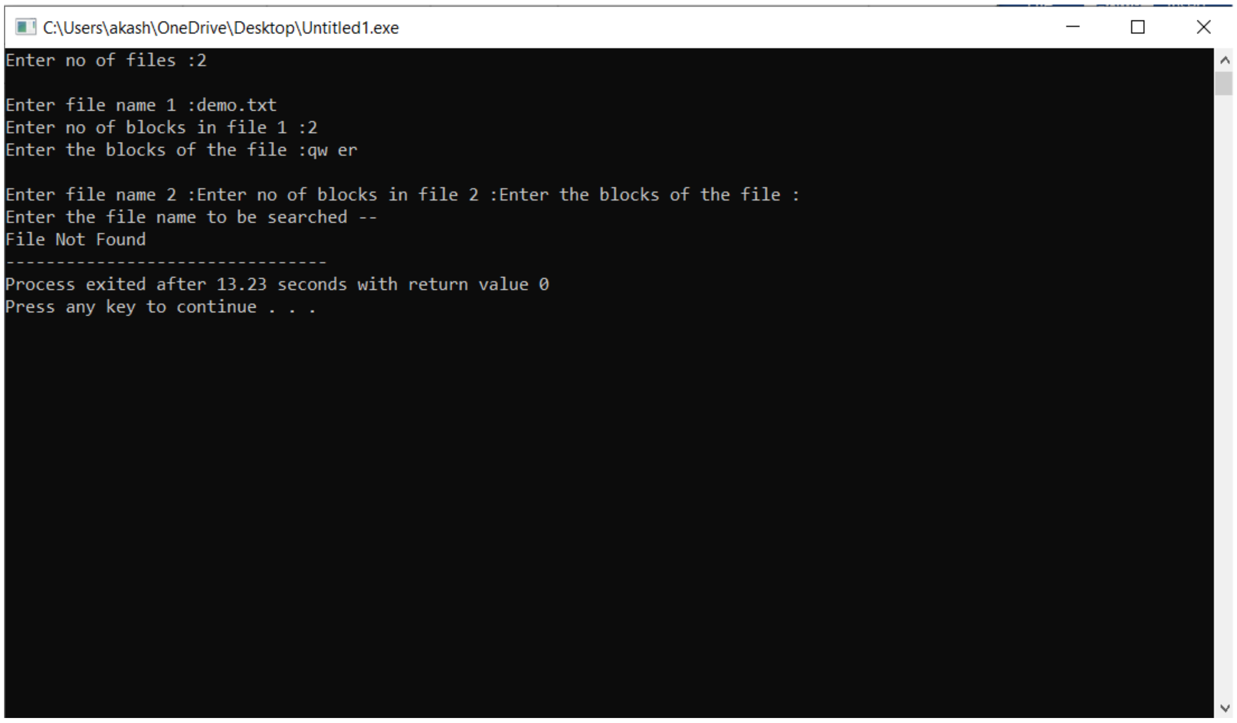
c. Linked: **Program**

```
#include<stdio.h> #include<conio.h>
```

```
struct fileTable
{
char name[20];
int nob; struct
block *sb;
}ft[30]; struct
block
{
int bno;
struct block *next;
};
void main()
{
int i, j, n;
char s[20];
struct block *temp;
clrscr(); printf("Enter no of
files :"); scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter file name %d :",i+1);
scanf("%s",ft[i].name); printf("Enter no of
blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
ft[i].sb=(struct block*)malloc(sizeof(struct block));
temp = ft[i].sb; printf("Enter the blocks of the file
:"); scanf("%d",&temp->bno); temp->next=NULL;
for(j=1;j<ft[i].nob;j++)
{
temp->next = (struct block*)malloc(sizeof(struct block));
temp = temp->next;
scanf("%d",&temp->bno);
}
temp->next = NULL;
}
printf("\nEnter the file name to be searched -- ");
scanf("%s",s); for(i=0;i<n;i++) if(strcmp(s,
ft[i].name)==0) break; if(i==n)
printf("\nFile Not Found");
else
{
```

```
printf("\nFILE NAME NO OF BLOCKS BLOCKS OCCUPIED");
printf("\n %s\t\t%d\t",ft[i].name,ft[i].nob);
temp=ft[i].sb; for(j=0;j<ft[i].nob;j++)
{
printf("%d ? ",temp->bno);
temp = temp->next;
}
}
getch();
}
```

OUTPUT



C:\Users\akash\OneDrive\Desktop\Untitled1.exe

```
Enter no of files :2

Enter file name 1 :demo.txt
Enter no of blocks in file 1 :2
Enter the blocks of the file :qw er

Enter file name 2 :Enter no of blocks in file 2 :Enter the blocks of the file :
Enter the file name to be searched --
File Not Found
--------------------------------
Process exited after 13.23 seconds with return value 0
Press any key to continue . . .
```