ARUNAI ENGINEERING COLLEGE

(An Autonomous institution)

VeluNagar,Thiruvannamalai606603 www.arunai.org

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BACHELOR OF TECHNOLOGY

2024-2025

FOURTH SEMESTER

AL3452 – OPERATING SYSTEM LAB

ARUNAI ENGINEERING COLLEGE

(An Autonomous institution)
TIRUVANNAMALAI–606603

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

Certified that this is a bonafide record of work done by Name

:

UniversityReg.No :

Semester :

Branch :

Year :

Staff-in-Charge **HeadoftheDepartment**

Submitted for the _____

Practical Examination held on _____

Internal Examiner **External Examiner**

EX.NO& Date	EXPERIMENTS	PAGE NO	SIGN
1&	Installation of Windows Operating System		
2A&	Study of UNIX Operating System		
2B&	UNIX COMMANDS		
2C&	Shell Programming		
3&	Process Management Using System Cells:Fork,Exit, Getpid, Wait, Close		
4&	Various CPU Scheduling Algorithm		
5&	Illustrate the Inter Process Communication(IPC) Strategy		
6&	Implementation of Semaphores		
7&	Implementation Deadlock Detection Algorithm		
8&	Bankers Algorithm for DeadLock Avoidance		
9&	Implementation of Thread in C		
10&	Implementation of Paging Technique of Memory Management		
11&	Implementation of Memory Allocation Methods for Fixed Partition		
12&	Implementation of Page Replacement Algorithm		
13&	Implementation of Various File Organization Techniques		
14&	Implementation of File Allocation Strategies		
15&	Implementation of VariousDisk Scheduling Algorithm		
16&	InstallanyGuessOperatingSystemDATE:likeLinux using VM Ware		

EX:1	Installation of WindowsOperating System
DATE:	

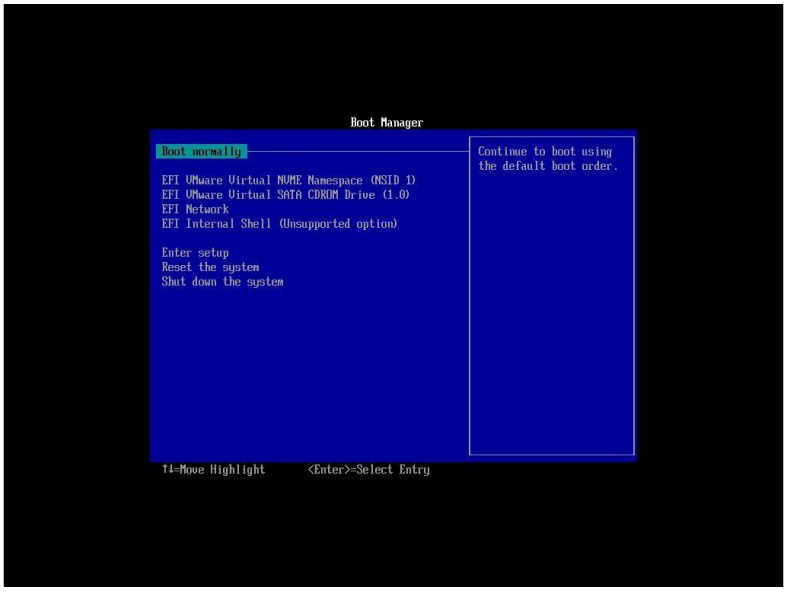
AIM:

ALGORITHM:

Steps:

1. BootyourPCusingWindows7DVD/USBdriveandpressany key if you see **Press any key tocontinue** message.

2. PresstheF8keyMultiple Timeswhilethecomputerstartstocall the Boot Manager Menu

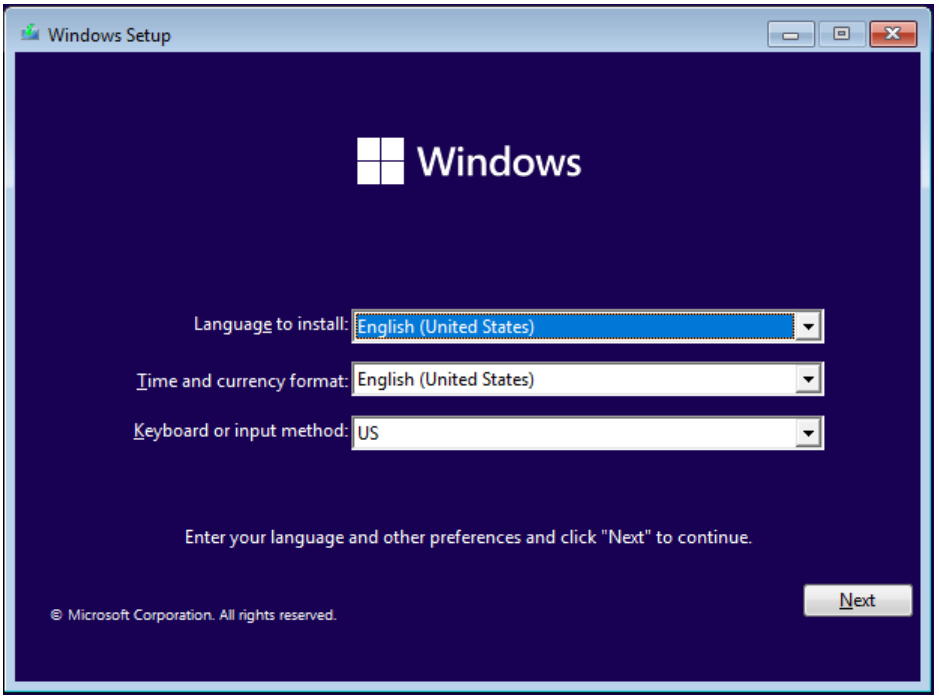


3. UsethearrowkeystoSelectthedrive unitthatcontainstheWindows11installer.

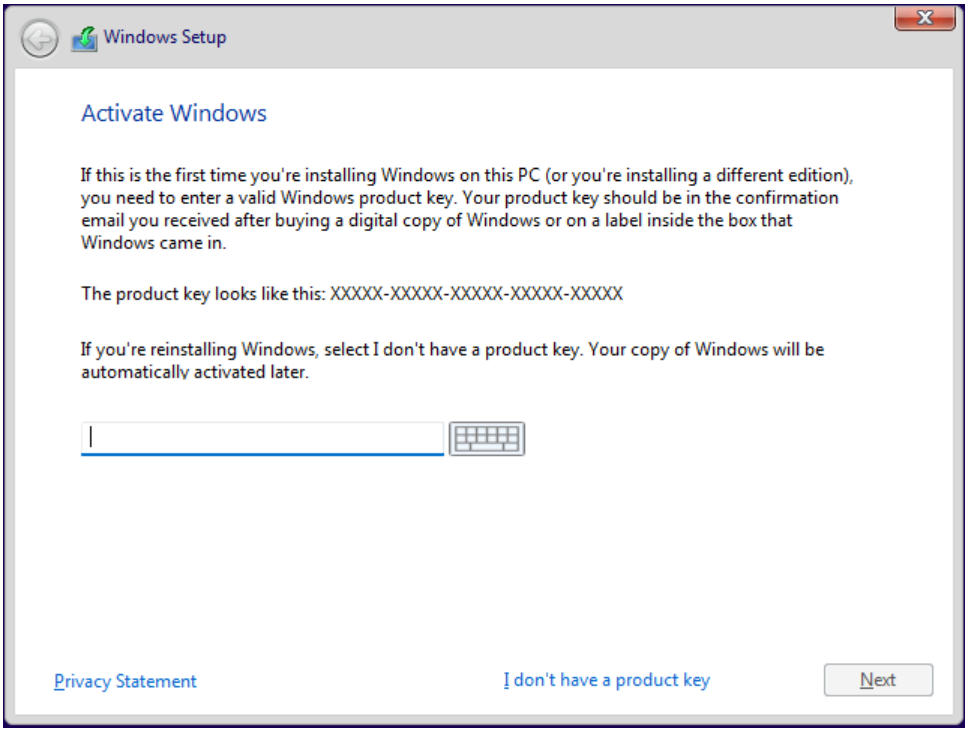
4. Pressanykeytoboot fromCDorDVD. Doingso itwillstarttheWindows11 Installation Process

5. NowtochooseInstallNow..

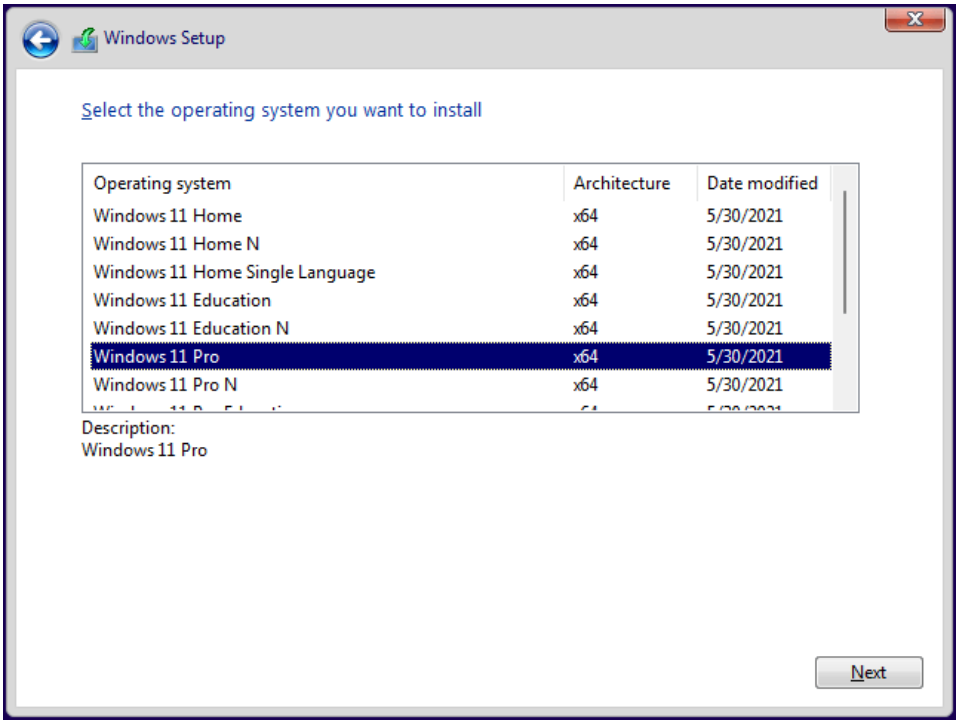
7. Specifythe language,time,andcurrencyformat,aswellaskeyboardorinput method to continue



8. Activate Windows. You are required to input your product key to go on. If you don't haveaproductkeyorforget,youcanactivateWindows11 later.Ifso,click I don't have a product key.



9. SelectwhicheditionofWindows11you'dliketoinstall,Windows11Home, Windows 11 Education, Windows 11 Pro…?

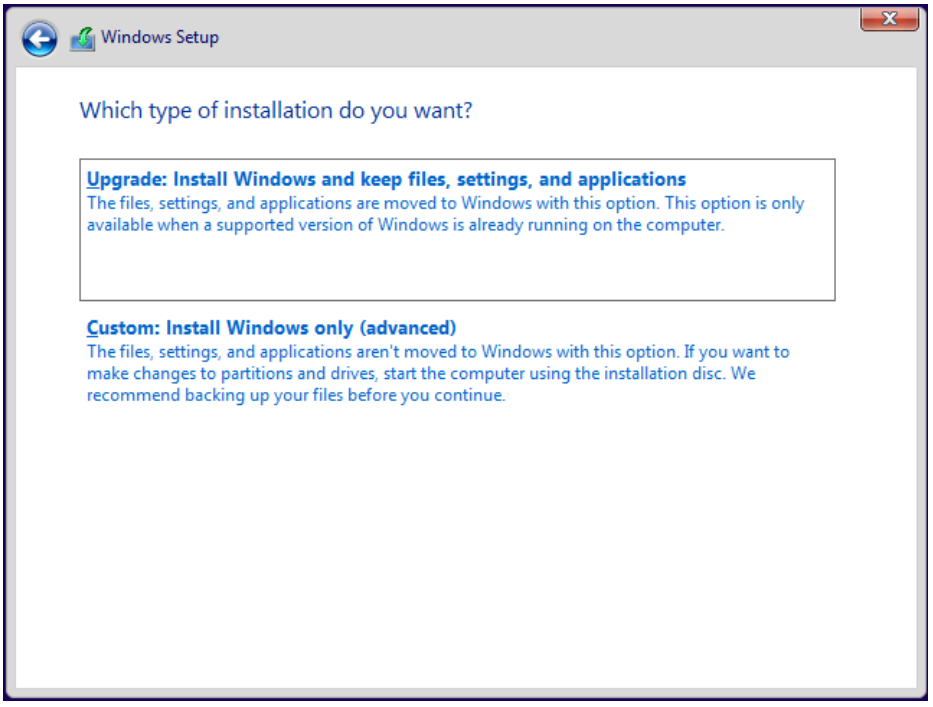


10. ClicktoacceptthеMicrosoftSoftware LicenseTermsandclickNexttogo on.

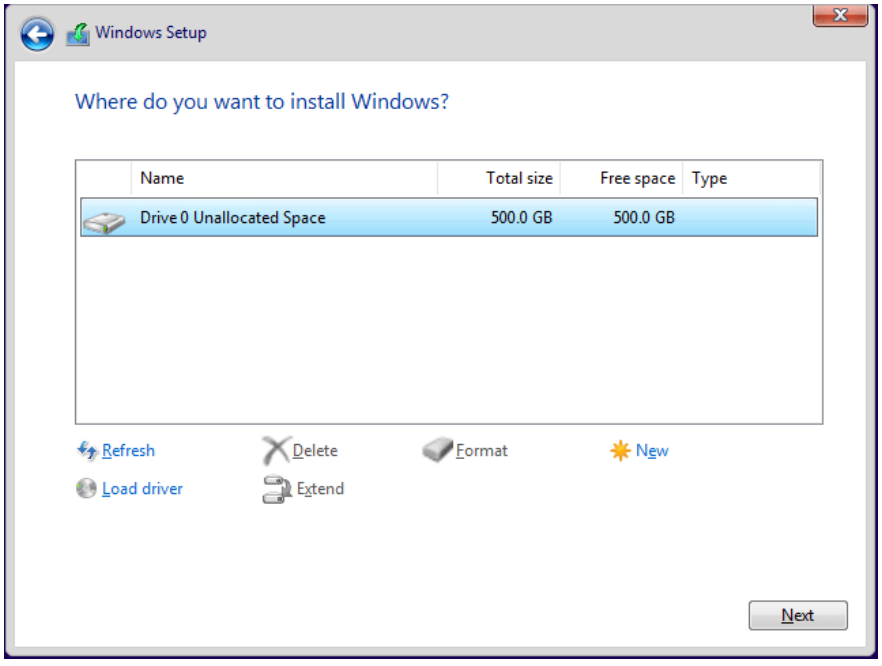
11. Whichtypeofinstallationdo youwant?Justchooseoneofthebelowtwo options according to your own situation.

Upgrade:InstallWindowsandkeepfiles,settings,andapplications.
The files,settings,andapplicationsare movedtoWindowswiththisoption.This option is only available when a supported version of Windows is already runningonthecomputer.

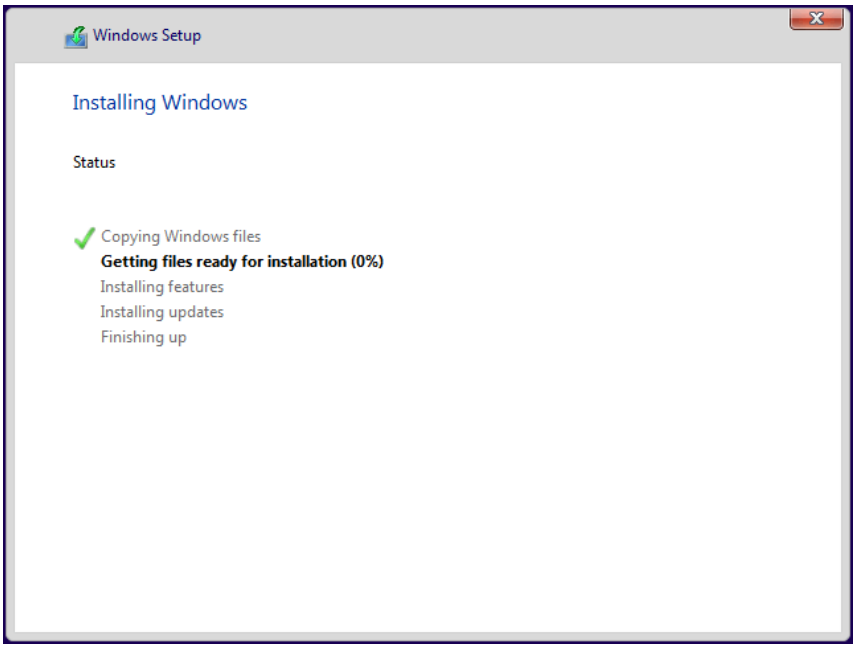
Custom:InstallWindowsonly(advanced). Thefiles,settings,andapplications aren't moved to Windows with this option. If you want to make changes to partitions and drives, start the computer using the installation disc.
Werecommendbackingupyourfilesbeforeyoucontinue.



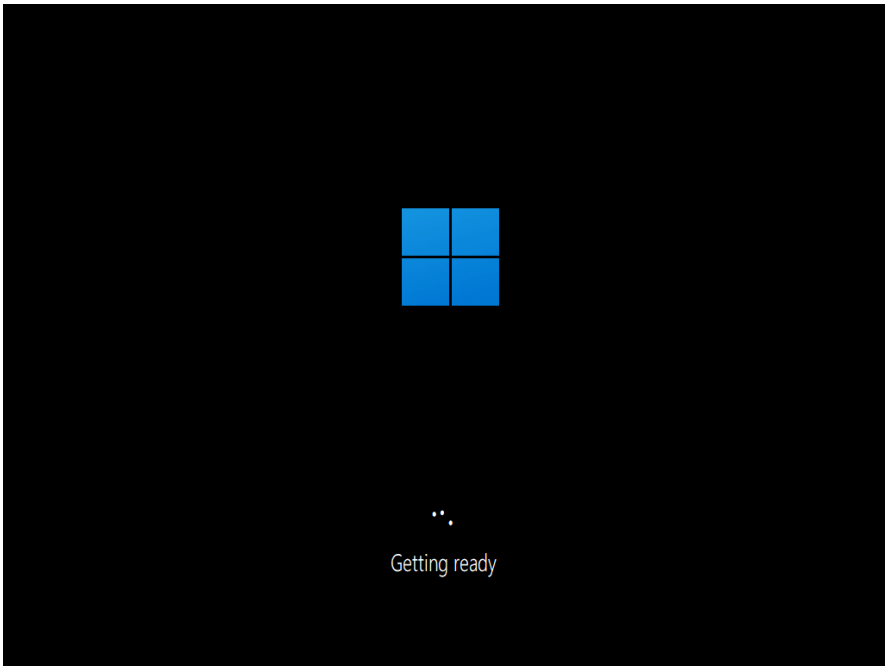
12. Wheredoyouwantto installWindows 11?Justpick upaharddiskto continue.



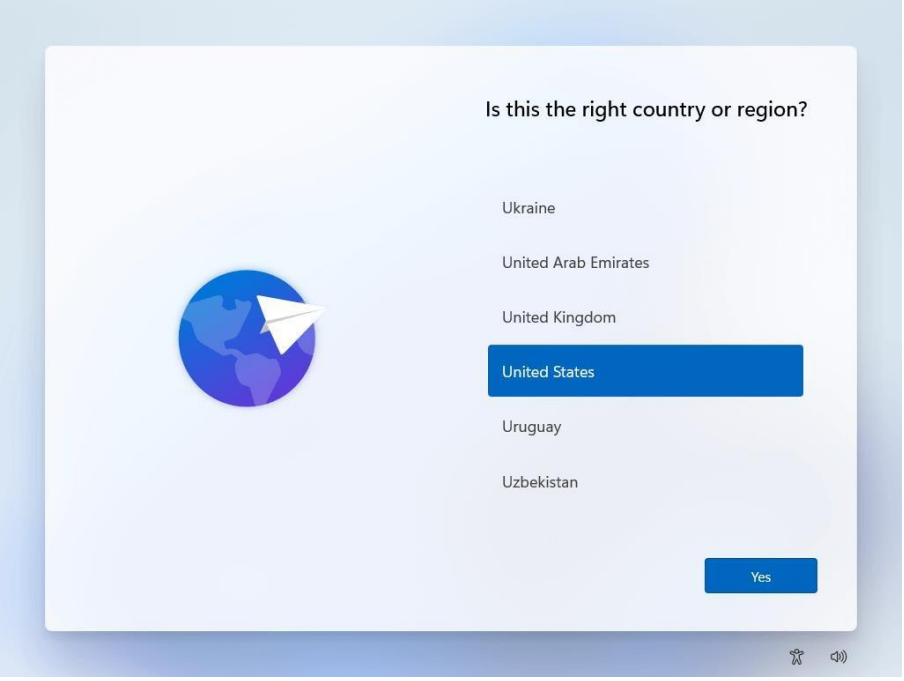
13. Waitpatientlyuntilitcompletestheinstallation.

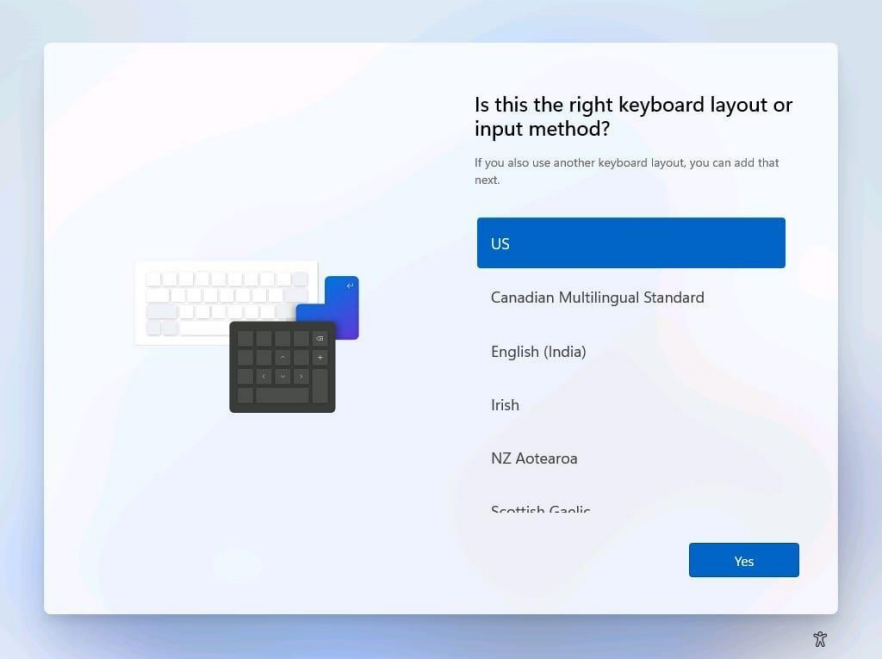


14. Then,youneedtowaitacoupleofminutestoallow Windows **11tostartsup**. During this period, you will first see **Windows 11 logo**.

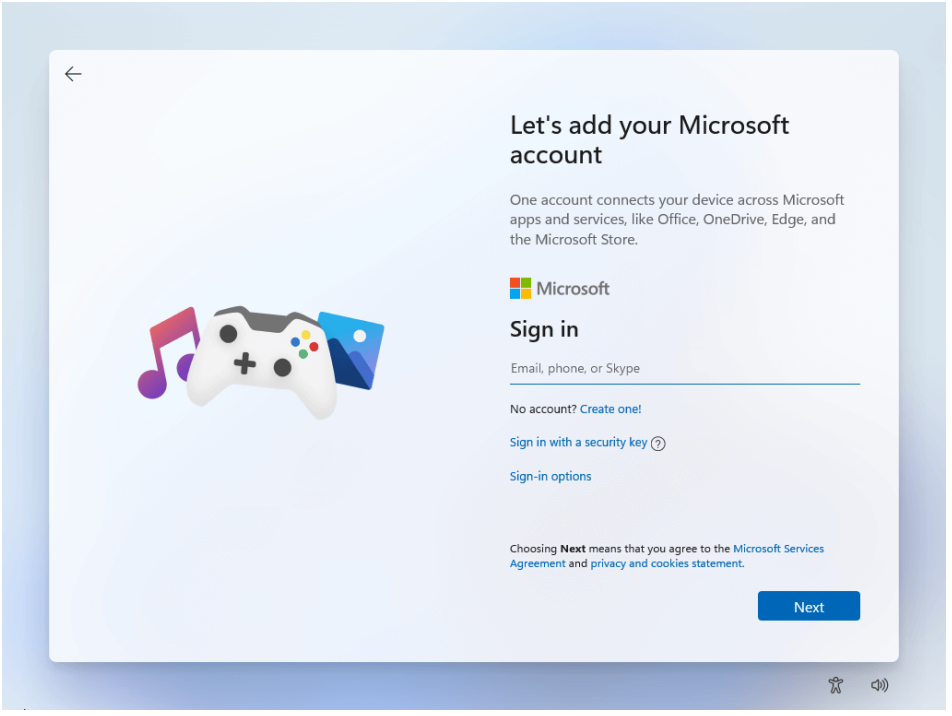


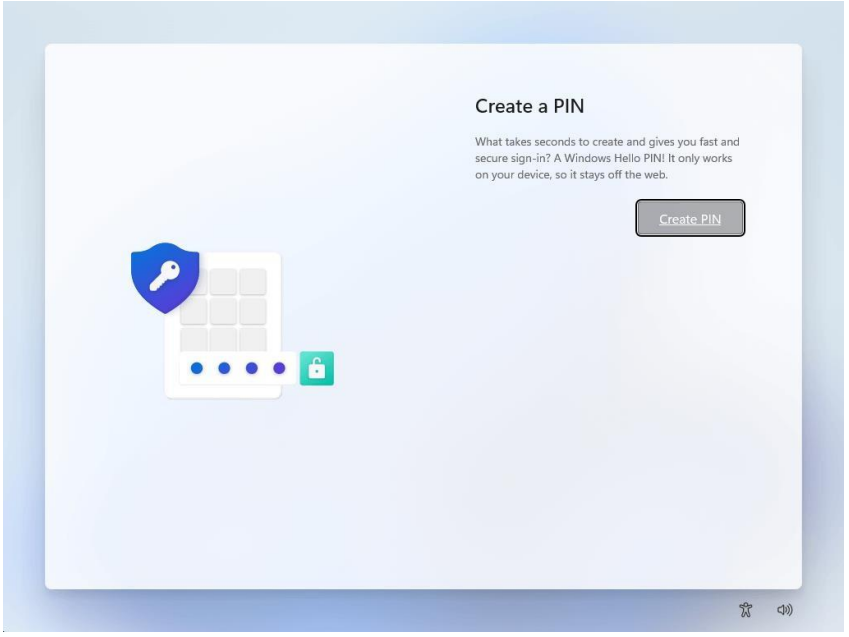
15. Then,youwillbeinvitedtocustomize your Windows11settings.Firstofall, customize your country or region.



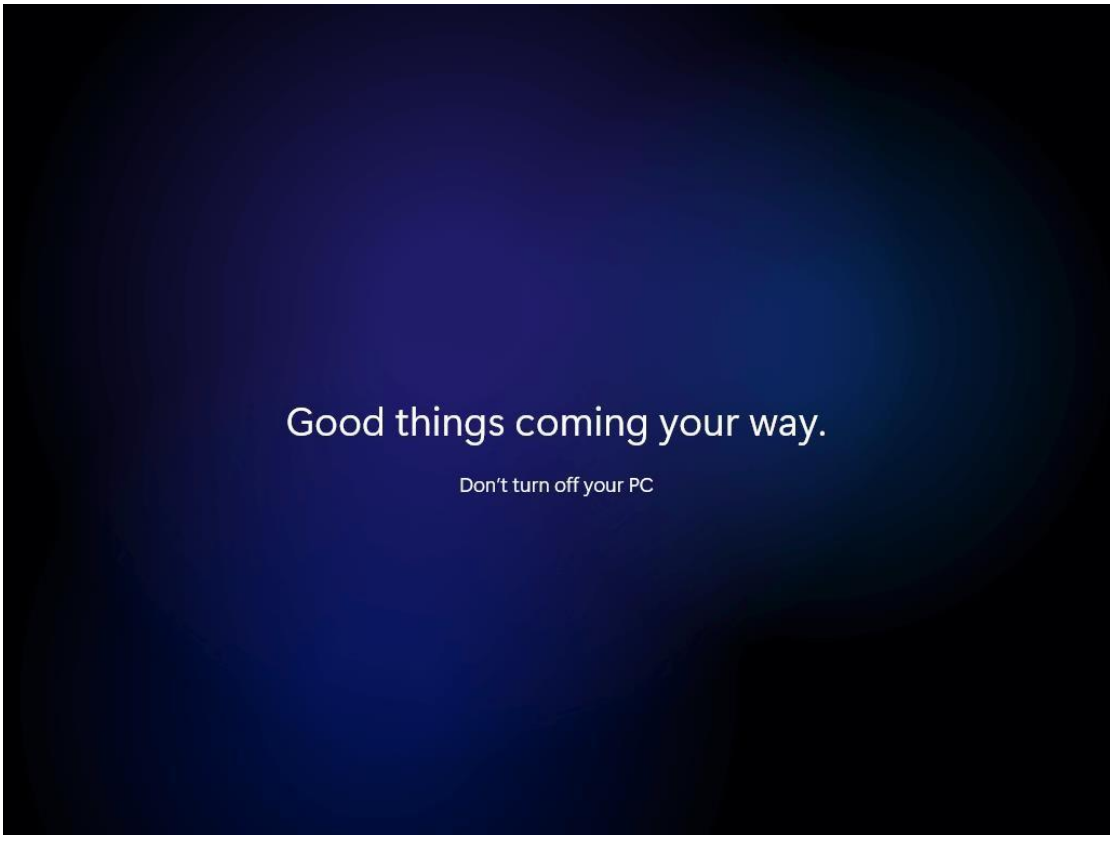


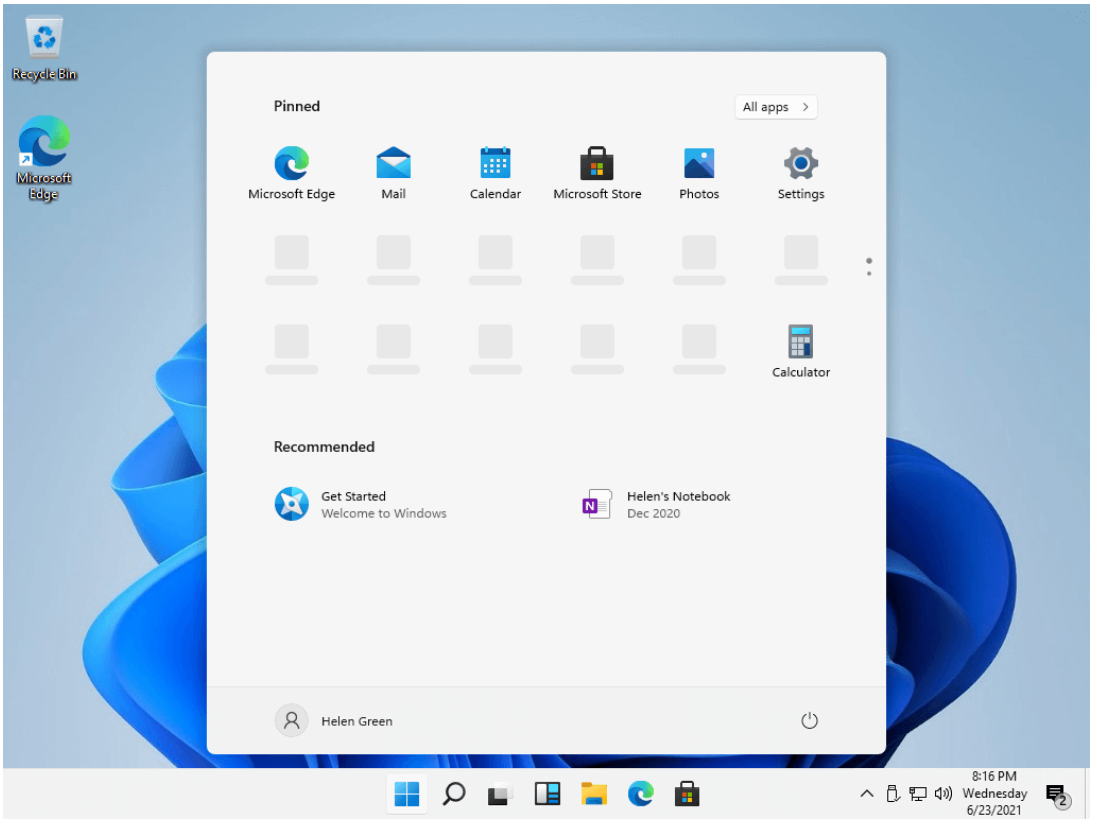
16. CreateMicrosoftAccountandPinNumber





17. WelcometoWindows11,StartExperiencingWindows11





Result:

EX:2A	**Studyof UNIX Operating System**
DATE:	

AIM:

ALGORITHM:

OPERATINGSYSTEM:

AnOperating Systemisasetofprogramsthat:

* Functionsasavirtualmachinebypresentinganinterfacethat iseasierto program than the underlying hardware

* Actsasresourcemanagement throughorderlyandcontrolledallocationofthe processors, memories,and I/O devices among the programs competing for it.

UNIXFeature:

1. Multi-user system - Multi-user capability of UNIX allows several users to use the same computerto performtheirtasks. Severalterminals [KeyboardsandMonitors]areconnectedto single powerful server.

2. Multi-taskingsystem- Multitasking isthecapabilityoftheoperatingsystemto perform various task simultaneously, i.e. a user can run multiple tasks concurrently.

3. ProgrammingFacility-theUNIXshellhasallthe necessaryingredientslikeconditionaland control structures, etc.

4. Security-Everyuser must haveasinglelogginnameandpassword.So,accessinganother user's data is impossible without his permission.

Apartfromthese features, UNIXhasanextensiveToolkit,exhaustivesystemcallsandLibraries and enhanced GUI (X Window).

OrganizationofUNIX:

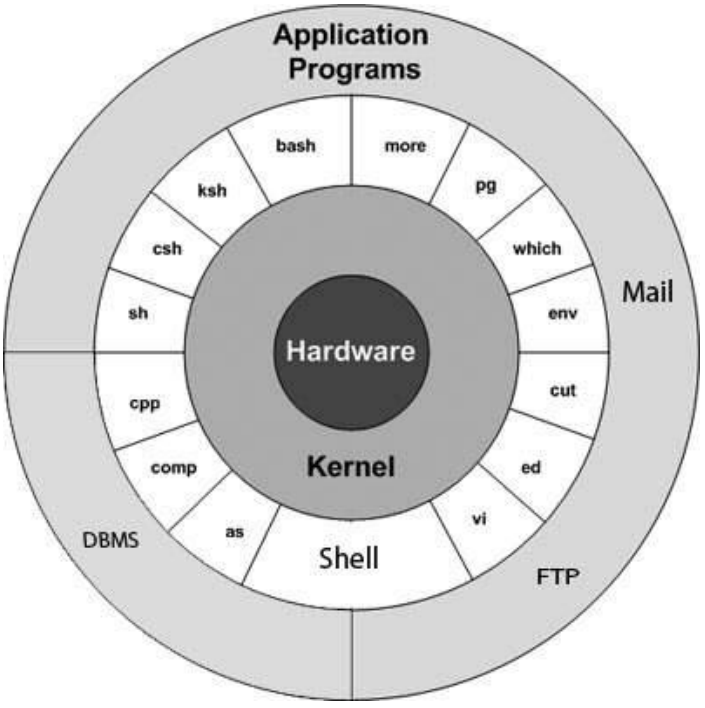
1. Thekernelistheheartofthesystem,acollectionofprogramswritten in C. Itmanagesthesystemresources,allocates time betweenuserandprocesses,decidesprocesspriorities and performs allother tasks. The kernel, in traditional parlance, is often called Operating system.

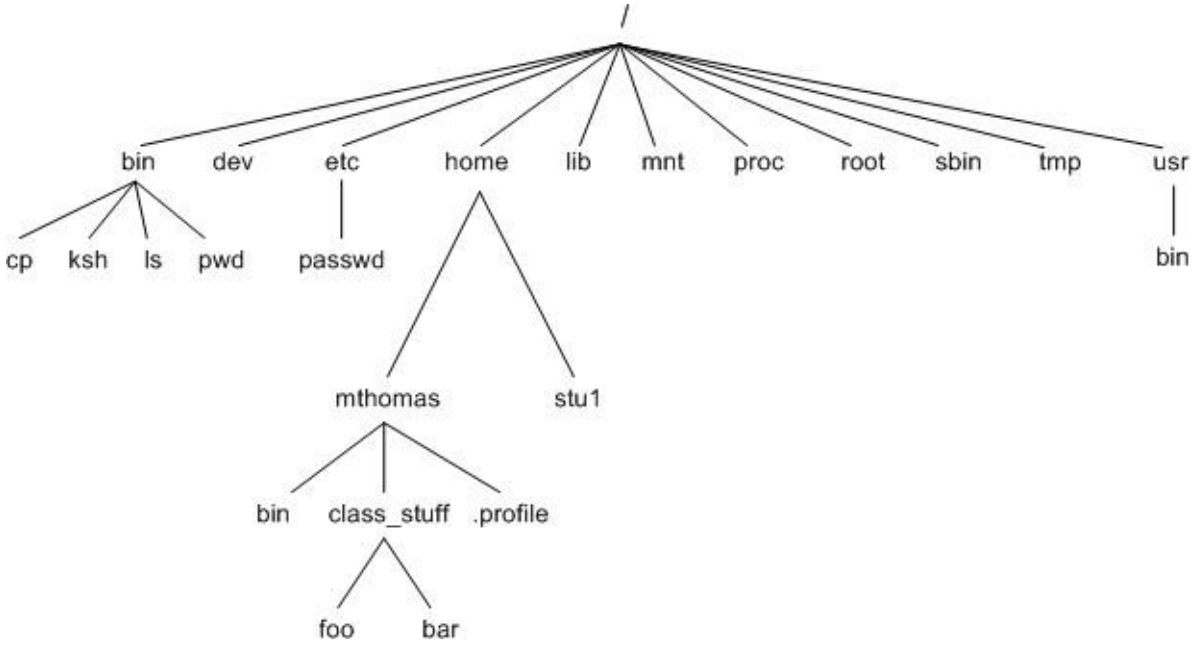
2. Theshell,ontheother hand,isthe"sleepingbeauty"ofUNIX.It is actually the interfacebetweentheuser andthekernel. Theshellistheagencywhichtakescareofthefeatures of redirection and has a programming capability of its own.

3. TheToolsandApplicationsconsistofApplicationSoftware,Compilers,DatabasePackage,Internet tools, UNIX commands, etc.

FileSystem:

All files in UNIX are related to one another. The file systemof UNIX resembles a tree that grows fromtoptobottomasshowninthe figure. The file systembeginswithadirectorycalledroot(atthe top). The root directory is denoted bya slash (\). Branching from root there are several directories such as bin, lib, etc, tmp, dev. Each of these directories contains severalsub-directories and files.





RESULT:

EX:2B	**UNIX COMMANDS**
DATE:	

AIM:

ALGORITHM:

LOGIN:

Typetelnetserver_ipaddressinrunwindow.

User mustauthenticatehimselfbyprovidingusernameandpassword.Onceverified,agreeting and $ prompt appears. The shell is now ready to receive commands from the user. Options suffixed with a hyphen (–) and arguments are separated by space

GENERALCOMMANDS:

COMMANDS	FUNCTIONS
Date	Usedto displaythecurrentsystemdateand time.
Date+%D	Displaysdateonly
Date+%T	Displaystimeonly
Date+%Y	Displaystheyearpartofdate
date+% H	Displaysthehourpartoftime
Cal	Calendar ofthecurrentmonth
calyear	Displayscalendarforallmonthsofthespecifiedyear
calmonthyear	Displayscalendar for thespecifiedmonthoftheyear
Who	LogindetailsofallserssuchastheirIP,TerminalNo, User name,
whoami	Usedtodisplaythelogindetailsoftheuser
Uname	DisplaystheOperatingSystem
uname–r	Showsversionnumber oftheOS (kernel).
uname–n	Displaysdomainnameoftheserver
echo$HOME	Displaystheuser'shomedirectory
bc	Basiccalculator.PressCtrl+dtoquit
lpfile	Allowstheuser tospoola jobalongwithothersina print queue
mancmdname	Manualforthegivencommand. Pressqtoexit
history	Todisplaythecommandsusedbytheuser sincelog on.
exit	Exitfromaprocess.Ifshellistheonlyprocessthen logs out

DIRECTORYCOMMANDS:

COMMANDS	FUNCTIONS
Pwd	Pathofthepresentworkingdirectory
mkdirdir	Adirectoryiscreatedinthegivennameunder the current directory
mkdirdir1 dir2	Anumberofsub-directoriescanbecreatedunderone stroke
cdsubdir	ChangeDirectory.Ifthesubdir startswith/thenpath starts fromroot (absolute) otherwise from current working directory
cd	Tomovebacktotheparentdirectory
rmdir subdir	Toswitchtotherootdirectory.
cd ..	Tomovebacktotheparentdirectory
rmdir subdir	Removesanemptysub-directory

FILECOMMANDS:

COMMANDS	FUNCTIONS
cat>filename	Tocreatea filewithsomecontents.Toendtypingpress Ctrl+d.The>symbolmeansredirectingoutputtoafile. (< for input).
catfilename	Displaysthefilecontents
cat>>filename	Usedtoappendcontents toafile
cp srcdes	Copyfilestogivenlocation.Ifalreadyexists, itwillbe overwritten
cp–isrcdes	Warnstheuser prior tooverwritingthedestinationfile
cp–rsrcdes	Copiestheentiredirectory,allitssub-directoriesand files
mvoldnew	Torenameanexistingfileor directory. –ioptioncan also be used
mvf1 f2f3 dir	Tomoveagroup offilestoadirectory
mv–voldnew	Displaynameofeachfileasit ismoved
rm file	Usedtodeletea fileor groupoffiles. –ioptioncanalso be used
rm*	Todeleteallthefilesinthedirectory
rm–r*	Deletesallfilesandsub-directories
rm–f*	Toforciblyremove evenwrite-protectedfiles
Ls	Listsallfilesandsubdirectories(bluecolored)insorted manner.
Is name	Tocheckwhether a fileordirectory exists
lsname*	Short-handnotationtolistoutfilenamesofaspecific pattern

COMMANDS	FUNCTIONS
ls–a	Listsallfilesincludinghiddenfiles(filesbeginning with .)
ls –xdirname	Tohavespecificlistingofadirectory.
ls–R	Recursivelisting ofallfilesinthesubdirectories
ls–l	Long listing showing file access rights (read/write/executerwxforuser/group/others-ugo)
cmpfile1file2	Usedtocomparetwofiles. Displaysnothingiffilesare identical
wcfile	itproducesastatisticsoflines(l),words(w),and characters(c).
chmodpermfile	Changespermissionforthespecifiedfile. (r=4, w=2, x=1)chmod740filesetsallrightsfor user,readonly for groups and no rights for others

Thecommandscanbecombinedusingthepipeline (|)operator. For

example:

numberofusers logged incanbeobtainedas. who |wc -lFinallytoterminatethe unix session execute the command exit or logout.

OUTPUT:

```
ThuMar2822:00:05IST2024
28/03/24
22:00:05
2024
28
March 2024

Su  Mo  Tu  We   Th   Fr   Sa
 1   2   3   4    5    6   7
 8   9  10  11   12   13  14
15  16  17  18   19 2021
22  23  24  25   26 2728
29  30  31

root:0Mar28 22:00Goku
pts/00Mar2822:00(scl-64) Linux
2.4.20-8smp
localhost.localdomain
/home/Goku
Goku
500+500
1000
/home/Goku/shellscripts/loops
list.sh
regexpr shellscripts
[Goku@localhost/]$
[Goku@localhost/]$cd/home/Goku/shellscripts/loops/$cd..
[Goku@localhost shellscripts]$
hicse
wishinguthebest hi
ece-a
wishinguthebest Bye
hicse
wishinguthebest
list.shregexpr shellscripts
..bash_logout        .canna      .gtkrc      regexpr      .viminfo.tmp
 ...bash_profile     .emacs      .kde        shellscripts .xemacs
   .bash_history     .bashrc     gree        list.sh      .viminfo
```

-rw-rw-r-- 1GokuGoku 32Apr11 14:52 greet-rw-
rw-r-- 1GokuGoku30Apr413:58list.sh
drwxrwxr-x2Goku Goku4096Apr 914:30regexpr

Greet list.sh regexpr shellscripts
demo greet
overwrite'greet'?n
$mvgreet greet.txt
$lsgreet.txtlist.shregexpr shellscripts
$mvgreet.txt ./regexpr/
$lslist.shregexprshellscripts
$rm-i*.shrm:removeregular file 'fact.sh'?
y rm: remove regular file 'prime.sh'? y
$lslist.shregexprshellscripts
$wc list.sh
4 9 30 list.sh
4list.sh
differ:byte1,line1
-rw-rw-r-- 1Goku Goku 30Apr4 13:58 list.sh

RESULT:

EX:2C	**Shell Programming**
DATE:	

AIM:

ALGORITHM:

INTRO:

The activities of a shell are not restricted to command interpretation alone. The shell also has rudimentaryprogramming features.Shellprogramsarestoredina file(withextension.sh).Shell programs run in interpretive mode. Bourne shell(sh), C shell(csh) and Kornshell(ksh) are also widelyused. Linux offers Bash shell (bash).

PRELIMINARIES:

1. Commentsinshellscriptstartwith#.

2. Shellvariablesare looselytyped i.e.notdeclared.Variablesinanexpressionoroutput must be prefixed by $.

3. Thereadstatementisshell'sinternaltoolformakingscriptsinteractive.

4. Outputisdisplayedusingechostatement.

5. Expressionsarecomputedusingtheexprcommand.Arithmeticoperatorsare+ -*/%.Meta characters * ( ) should be escaped with a \.

6. Theshellscriptsareexecuted$sh filename

DECISIONMAKING:

//Ifelse

if[condition]then

statements

else

statements

fi

//elseif

if[condition]then

statements

elif[condition]then

statements

else

statements

fi

Thesetofrelationaloperatorsare –eq–ne–gt–ge–lt –leand logicaloperatorsusedinconditional expression are –a –o !

MULTI-WAY BRANCHING:

The case statement is used to compare a variables value against a set ofconstants. If it matches aconstant,thenthesetofstatementsfollowedafter)isexecutedtilla ;;isencountered. Theoptional default block is indicated by*. Multiple constantscan be specified ina single patternseparatedby|. case variable in constant1)

statements;;

constant2)

statements;;

...*)

Statements

Esac

LOOPS:

Shellsupportsaset ofloopssuchasfor,whileanduntil toexecutea set ofstatementsrepeatedly. Thebodyofthe loop is contained between do and done statement.

```
//forloop
forvariableinlistdo
 statements done

//while loop
while[condition]do
 statements done

// until condition
until[condition]do
statements done
```

PROGRAMS:

(A) SWAPPING

```
var1="Hello"
var2="World"

echo"Beforeswapping:"
echo "var1 = $var1"
echo "var2 = $var2"

temp=$var1
var1=$var2
var2=$temp

echo"Afterswapping:"
echo "var1 = $var1"
echo "var2 = $var2"
```

(B) FahrenheittoCentigradeVersion

```
fahrenheit_to_celsius() {
celsius=$(echo "scale=2;($1 -32)*5/9"|bc) echo
"$celsius"
}
echo"EntertemperatureinFahrenheit:"
read fahrenheit
celsius=$(fahrenheit_to_celsius "$fahrenheit")
echo"$fahrenheitFahrenheitis$celsiusCelsius"
```

(C) BIGGESTOFTHREE NUMBER:

```
find_max(){
max=$1
if[$2-gt$max];then
max=$2
fi
if[$3-gt$max];then
max=$3
fi
echo"$max"
}
echo"Enterthreenumbersseparatedbyspaces:"
read num1 num2 num3
max=$(find_max$num1$num2$num3)
echo "The largest number is: $max"
```

(D) GRADEDETERMINATION

```
determine_grade(){
  score=$1
grade=""
if[$score-ge90];then
grade="A"
elif[$score-ge80];then
grade="B"
elif[$score-ge70];then
grade="C"
elif[$score-ge60];then
grade="D"
else
grade="F"
fi
echo"$grade"
}
echo"Enterthenumericalscore:"
read score
grade=$(determine_grade$score)
echo"Thegradeforthescore$scoreis:$grade"
```

(E) VOWELORCONSONANT:

```
is_vowel(){
case"$1"in
[aeiouAEIOU])echo"Vowel";;
*)echo"Consonant";;
esac
}
echo"Enteracharacter:"
read character
result=$(is_vowel"$character")
echo"Thecharacter $characterisa$result."
```

(F) SIMPLECALCULATOR

```
add(){
 result=$(echo"$1+$2"|bc)
 echo "$result"
}
subtract(){
result=$(echo"$1-$2"|bc)
echo "$result"
}
multiply(){
result=$(echo"$1*$2"|bc) echo
"$result"
}
divide(){
result=$(echo"scale=2;$1/$2"|bc)
echo "$result"
}
echo"SimpleCalculator"
echo"Availableoperations:+, -,*,/"
echo "Enter operation (e.g., 5 + 3):"
read num1 operator num2
case"$operator"in
 "+")result=$(add$num1$num2);;
"-") result=$(subtract $num1 $num2);;
"*")result=$(multiply$num1$num2);; "/")
result=$(divide $num1 $num2);;
 *)echo"Invalidoperator";exit1;; esac
echo"Result:$result"
```

(G) MULTIPLICATION

```
multiplication_table(){
num=$1
echo"Multiplicationtablefor$num:"
for (( i=1; i<=10; i++ )); do
result=$((num * i))
echo"$numx$i=$result"done
}
echo"Enteranumbertogenerateits multiplicationtable:"
read number
multiplication_table$number
```

(H) NUMBERREVERSE

```
reverse_number(){
num=$1
reversed=""
while [ $num -gt 0 ]; do
digit=$((num % 10))
reversed="${reversed}${digit}"n
um=$((num / 10))
 done
echo"$reversed"
}
echo"Enteranumber:"
read number
reversed=$(reverse_number $number)
echo"Thereversednumberis:$reversed"
```

(I) PRIMENUMBER

```
is_prime(){
num=$1
if[$num-le1];then
echo"$numisnotaprimenumber."return
fi
for((i=2;i*i<=num;i++));do if [
 $((num % i)) -eq 0 ]; then
echo"$numisnotaprimenumber."return
fi
done
echo"$numisaprime number."
}
echo"Enteranumbertocheckifit'sprime:" read
number

is_prime$number
```

OUTPUT:

(A)

Beforeswapping:
var1 = Hello
var2 = World
Afterswapping:
var1 = World
var2 = Hello

(B)

EntertemperatureinFahrenheit: 32
32Fahrenheitis0.00Celsius

(C)

Enterthreenumbersseparatedbyspaces: 5
12 8
Thelargestnumber is:12

(D)

Enterthenumericalscore:
85
Thegradeforthescore85is: B

(E)

Enteracharacter:
a
Thecharacter aisaVowel.

Enteracharacter:
b
Thecharacterbisa Consonant.

(F)

SimpleCalculator
Availableoperations:+,-,*,/
Enter operation (e.g., 5 + 3):
5+3
Result:8

(G)

Enteranumbertogenerateitsmultiplicationtable: 5
Multiplicationtablefor5: 5
x 1 = 5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50

(H)

Enteranumber:
12345
Thereversednumberis:54321

(I)

Enteranumbertocheckifit'sprime: 17
17isaprimenumber.

Enteranumbertocheckifit isprime: 20
20isnotaprimenumber.

RESULT:

EX:3	Process Management Using SystemCells:Fork, Exit, Getpid, Wait, Close
DATE:	

AIM:

ALGORITHM:

PROGRAM:

(A) (Fork,Getpid,Exit)

```
#include <stdio.h>
#include <unistd.h>
#include<sys/types.h>
#include <sys/wait.h>

intmain(){
pid_tpid=fork();

if(pid<0)
  printf(stderr,"Forkfailed\n");
return 1;
  }elseif(pid== 0){
   printf("Childprocess:PID=%d\n",getpid());
   printf("Child process exiting\n");
   exit(0);
} else{
   printf("Parentprocess:PID=%d\n",getpid());
wait(NULL);
   printf("Parentprocessexiting\n");
   }
   return0;
}
```

(B) (waitsystemcall)

```
#include <stdio.h>
#include <stdlib.h>
#include<sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

intmain(){
pid_tpid=fork();
if (pid < 0){
perror("fork");
 exit(EXIT_FAILURE);
 }elseif(pid==0){
 printf("Childprocess:PID=%d\n",getpid());
 sleep(2);
 printf("Childprocessexiting\n");
 exit(EXIT_SUCCESS);
}
else{

printf("Parent process: PID = %d\n", getpid());
printf("Waitingforchildprocesstofinish...\n"); int
status;
wait(&status);
 if(WIFEXITED(status)){
printf("Childprocessexitedwithstatus:%d\n",WEXITSTATUS(status));
}elseif(WIFSIGNALED(status)){
printf("Childprocessterminatedbysignal:%d\n", WTERMSIG(status));
}
printf("Parentprocessexiting\n");
}

return0;
}
```

(C) Open&closesystemcall

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include<unistd.h>

intmain(){
 intfd=open("example.txt",O_CREAT |O_WRONLY|O_TRUNC,0644); if
 (fd == -1) {
 perror("open");
 exit(EXIT_FAILURE);
  }
constchar*buffer="Hello,world!\n";
 ssize_tbytes_written=write(fd,buffer,strlen(buffer)); if
(bytes_written == -1) {
perror("write");close(fd);
  exit(EXIT_FAILURE);
}
 printf("Datawrittensuccessfullytothefile\n"); if
(close(fd) == -1) {
perror("close");
exit(EXIT_FAILURE);
}
printf("Fileclosedsuccessfully\n");
return 0;
}
```

OUTPUT:

(A)

Parentprocess:PID=1234
Child process: PID = 1235
Child process exiting
Parent process exiting

(B)

Parentprocess:PID=1234 Child
process: PID = 1235
Waitingforchildprocesstofinish...
Child process exiting
Childprocessexitedwithstatus:0
Parent process exiting

(C)

Datawrittensuccessfullytothefile File
closed successfully

RESULT:

EX 4	**Various CPU Scheduling Algorithm**
DATE:	

AIM:

ALGORITHM:

(A) FCFS Scheduling

Process Scheduling:

CPU scheduling is used in multiprogrammed operating systems.

By switching CPU among processes, efficiency of the system can be improved. Some scheduling algorithms are FCFS, SJF, Priority, Round-Robin, etc. Gantt chart provides a way of visualizing CPU scheduling and enables to understand better.

First Come First Serve (FCFS):

Process that comes first is processed first FCFS scheduling is non-preemptive Not efficient as it results in long average waiting time. Can result in starvation, if processes at beginning of the queue have Long bursts.

PROGRAMS:

```
#include<stdio.h>
struct Process {
   int id;        // Process IDint
   arrival;        //Arrivaltime
   int burst;      // Burst time
};
voidcalculateWaitingTime(structProcessproc[],intn, intwt[]){ wt =
      (int *)malloc(n * sizeof(int));
   if(wt==NULL){
      // Handle memory allocation failure
      printf("Memoryallocationfailed\n");
      return;
   }
   wt[0]=0;
   for(int i=1;i<n; i++){
      wt[i]= wt[i-1]+proc[i- 1].burst;
   }
      free(wt);
}
voidcalculateTurnaroundTime(structProcessproc[],intn,intwt[],inttat[]){ for (int
   i = 0; i < n; i++) {
   tat[i]=proc[i].burst+wt[i];
   }
}
voidcalculateAverageTime(structProcessproc[],intn){ int
   wt[n], tat[n], total_wt = 0, total_tat = 0;
   calculateWaitingTime(proc, n, wt);
   calculateTurnaroundTime(proc, n, wt, tat);
   printf("Process\tArrivalTime\tBurstTime\tWaitingTime\tTurnaroundTime\n"); for
   (int i = 0; i < n; i++) {
   total_wt+=wt[i];
   total_tat+=tat[i];
    printf("%d\t%d\t\t%d\t\t%d\t\t%d\n",proc[i].id,proc[i].arrival,proc[i].burst,wt[i],
tat[i]);
   }
   printf("\nAverage waiting time: %.2f\n", (float)total_wt / n);
   printf("Averageturnaroundtime:%.2f\n",(float)total_tat/n);
}
intmain(){
   structProcessproc[]={{1,0, 6}, {2,3,8},{3,5, 7},{4, 9, 3}};
   intn=sizeof(proc)/sizeof(proc[0]);
   calculateAverageTime(proc, n);
```

```
return0;
}
```

(B) SJF Scheduling

```
#include<stdio.h>
struct Process {int
id;
intarrival;
int burst;
intcompletion;
intturnaround;
int waiting;
};

voidswap(structProcess*a,structProcess*b){
struct Process temp = *a;
*a =*b;
*b= temp;
}

voidsortByArrival(structProcessproc[],intn){ for
(int i = 0; i < n - 1; i++) {
 for(intj=0;j< n-i-1;j++) {
 if(proc[j].arrival>proc[j+1].arrival){
 swap(&proc[j], &proc[j + 1]);
}
}
}
}
voidsortByBurst(structProcessproc[],intn){ for
(int i = 0; i < n - 1; i++) {
for(intj=0;j<n- i-1;j++) {
if(proc[j].burst>proc[j+1].burst){
swap(&proc[j], &proc[j + 1]);
}
}
}
}

voidcalculateCompletionTime(structProcessproc[],intn){ int
currentTime = proc[0].arrival;
for (int i=0; i<n; i++){
 if(currentTime<proc[i].arrival){ currentTime
= proc[i].arrival;
}
proc[i].completion=currentTime+proc[i].burst;
currentTime = proc[i].completion;
}
}

voidcalculateTurnaroundTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
 proc[i].turnaround=proc[i].completion-proc[i].arrival;
}
}

voidcalculateWaitingTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
proc[i].waiting=proc[i].turnaround-proc[i].burst;
```

```
                    }
                    }

                    voidcalculateAverageTime(structProcessproc[],intn){ int
                     totalWaiting = 0, totalTurnaround = 0;

                    for (int i = 0; i < n; i++) {
                     totalWaiting+=proc[i].waiting;
                     totalTurnaround+=proc[i].turnaround;
                    }
                     printf("Average Waiting Time: %.2f\n", (float)totalWaiting / n);
                     printf("AverageTurnaroundTime:%.2f\n",(float)totalTurnaround/n);
                    }
                    voidprintProcessDetails(structProcessproc[],intn){
                    printf("Process\tArrivalTime\tBurstTime\tCompletionTime\tTurnaroundTime\tWaiting
                    Time\n");
                    for(inti=0;i<n;i++){
                    printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",proc[i].id,proc[i].arrival,proc[i].burst,
                    proc[i].completion, proc[i].turnaround, proc[i].waiting);
                    }
                    }
                    intmain(){
                     structProcessproc[]={{1,0,6},{2,3,8}, {3,5,7},{4,9,3}};
                    intn=sizeof(proc)/sizeof(proc[0]);
                    sortByArrival(proc, n);
                    sortByBurst(proc, n);
                    calculateCompletionTime(proc, n);
                    calculateTurnaroundTime(proc, n);
                    calculateWaitingTime(proc, n);
                    printProcessDetails(proc, n);
                    calculateAverageTime(proc, n);
                    return 0;
}
```

(C) PriorityScheduling

```
#include<stdio.h>
struct Process {int
id;
int arrival;
int burst;
intpriority;
intcompletion;
int  turnaround;
int waiting;
};

voidswap(structProcess *a,structProcess*b){
struct Process temp = *a;
*a =*b;
*b= temp;
}
voidsortByArrival(structProcessproc[],intn){ for
(int i = 0; i < n - 1; i++) {
for(intj=0;j<n- i-1;j++) {
if(proc[j].arrival>proc[j+1].arrival){
swap(&proc[j], &proc[j + 1]);
}
}
}
}

voidsortByPriority(structProcessproc[],intn){ for
(int i = 0; i < n - 1; i++) {
for(intj=0;j<n- i-1;j++) {
 if(proc[j].priority>proc[j+1].priority){
swap(&proc[j], &proc[j + 1]);
 }
}
}
}
voidcalculateCompletionTime(structProcessproc[],intn){ int
currentTime = proc[0].arrival;
for (int i=0; i<n; i++){
if(currentTime<proc[i].arrival){ currentTime
= proc[i].arrival;
}
proc[i].completion=currentTime+proc[i].burst;
 currentTime = proc[i].completion;
}
}

voidcalculateTurnaroundTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
proc[i].turnaround=proc[i].completion-proc[i].arrival;
}
}
```

```
voidcalculateWaitingTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
proc[i].waiting=proc[i].turnaround-proc[i].burst;
 }
}
voidcalculateAverageTime(structProcessproc[],intn){ int
totalWaiting = 0, totalTurnaround = 0;
for (int i = 0; i < n; i++) {
 totalWaiting+=proc[i].waiting;
totalTurnaround+=proc[i].turnaround;
}
printf("Average Waiting Time: %.2f\n", (float)totalWaiting / n);
printf("AverageTurnaroundTime:%.2f\n",(float)totalTurnaround/n);
}
voidprintProcessDetails(structProcessproc[],intn){
 printf("Process\tArrivalTime\tBurstTime\tPriority\tCompletionTime\tTurnaround
Time\tWaiting Time\n");
for (int i=0; i<n; i++){
printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",proc[i].id,proc[i].arrival,proc[i].burst,
proc[i].priority, proc[i].completion, proc[i].turnaround, proc[i].waiting);
 }
}
intmain(){
 structProcess proc[]={{1,0,6,2},{2,2,8, 1},{3,3,7,3},{4,5, 3,4}};
 intn=sizeof(proc)/sizeof(proc[0]);

sortByArrival(proc, n);
sortByPriority(proc, n);
calculateCompletionTime(proc,n);
calculateTurnaroundTime(proc,n);
calculateWaitingTime(proc, n);
printProcessDetails(proc, n);
calculateAverageTime(proc, n);

 return0;
}
```

(D) RoundRobinScheduling

```
#include<stdio.h>
#defineTIME_QUANTUM2
struct Process {
intid;
intarrival;
int burst;
int remaining;
intcompletion;
intturnaround;
int waiting;
};

voidswap(structProcess*a,structProcess*b){
struct Process temp = *a;
*a =*b;
*b= temp;
}
voidsortByArrival(structProcessproc[],intn){ for
(int i = 0; i < n - 1; i++) {
 for(intj=0;j< n-i-1;j++) {
if(proc[j].arrival>proc[j+1].arrival){ swap(&proc[j],
&proc[j + 1]);
}
}
}
}

voidcalculateCompletionTime(structProcessproc[],intn){ int
currentTime = 0;
intremainingProcesses=n;

while(remainingProcesses>0){ for
(int i = 0; i < n; i++) {
if(proc[i].remaining>0){
if(proc[i].remaining>TIME_QUANTUM){
currentTime += TIME_QUANTUM;
proc[i].remaining -= TIME_QUANTUM;
} else{
currentTime+=proc[i].remaining;
proc[i].remaining = 0;
proc[i].completion=currentTime;
remainingProcesses--;
}
}
}
}
}

voidcalculateTurnaroundTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
proc[i].turnaround=proc[i].completion-proc[i].arrival;
}
```

```
}
voidcalculateWaitingTime(structProcessproc[],intn){ for
(int i = 0; i < n; i++) {
proc[i].waiting=proc[i].turnaround-proc[i].burst;
}
}

voidcalculateAverageTime(structProcessproc[],intn){ int
totalWaiting = 0,
 totalTurnaround=0;

for (int i = 0; i < n; i++) {
totalWaiting+=proc[i].waiting;
totalTurnaround+=proc[i].turnaround;
}

printf("Average Waiting Time: %.2f\n", (float)totalWaiting / n);
printf("AverageTurnaroundTime:%.2f\n",(float)totalTurnaround/n);
}

voidprintProcessDetails(structProcessproc[],intn){
printf("Process\tArrivalTime\tBurstTime\tCompletionTime\tTurnaroundTime\tWaitingTime\n"); for
(int i = 0; i < n; i++) {
printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",proc[i].id,proc[i].arrival,proc[i].burst,
proc[i].completion, proc[i].turnaround, proc[i].waiting);
}
}

intmain(){
structProcess proc[]={{1,0,5,5},{2,1,3,3},{3,2,8,8},{4,3,6,6}};
intn=sizeof(proc)/ sizeof(proc[0]);

sortByArrival(proc, n);
calculateCompletionTime(proc,n);
calculateTurnaroundTime(proc,n);
calculateWaitingTime(proc, n);
printProcessDetails(proc, n);
calculateAverageTime(proc, n);

return0;
}
```

OUTPUT:

(A)

Process	ArrivalTime	Burst Time	WaitingTime	TurnaroundTime
1	0	6	0	6
2	3	8	6	14
3	5	7	14	21
4	9	3	21	24

Average waiting time: 10.25
Averageturnaroundtime:16.25

(B)

Process	ArrivalTime	Burst Time	CompletionTime	TurnaroundTime	WaitingTime
1	0	6	6	6	0
3	5	7	13	8	1
2	3	8	21	18	10
4	9	3	24	15	12

Average Waiting Time: 5.75
AverageTurnaroundTime:11.75

(C)

Process	ArrivalTime	Burst Time	Priority	CompletionTime	TurnaroundTime	WaitingTime
1	0	6	2	6	6	0
2	2	8	1	14	12	4
3	3	7	3	21	18	11
4	5	3	4	24	19	16

Average Waiting Time: 7.75
AverageTurnaroundTime:13.75

(D)

Process	ArrivalTime	Burst Time	CompletionTime	TurnaroundTime	WaitingTime
1	0	5	16	16	11
2	1	3	9	8	5
3	2	8	20	18	10
4	3	6	22	19	13

Average Waiting Time: 9.75
AverageTurnaroundTime:15.25

RESULT

EX:5	**Illustrate the Inter Process Communication(IPC)Strategy**
DATE:	

AIM:

ALGORITHM:

PROGRAM:

DESCRIPTION:

Inter ProcessCommunication(IPC)isamechanismthatinvolvescommunicationofone process with another process. This usually occurs only in one system.

Communicationcanbeoftwotypes:

1. Betweenrelatedprocessesinitiating fromonlyoneprocess,suchasparent andchild processes.
2. Betweenunrelated processes,ortwoormoredifferentprocesses.
3. Followingaresome importanttermsthat weneedtoknowbeforeproceeding further on this topic

Pipes − Communicationbetweentworelatedprocesses. The mechanism is half duplex meaningthefirst process communicateswiththesecondprocess. Toachievea full duplexi.e., for thesecondprocess tocommunicatewith the first process another pipe is required.

FIFO −Communicationbetweentwounrelatedprocesses. FIFOisa fullduplex, meaningthefirstprocesscan communicate with the second process and vice versa at the same time

MessageQueues − Communicationbetweentwo or moreprocesses withfullduplex capacity. Theprocesses will communicatewitheachother bypostinga messageandretrievingit out ofthequeue. Onceretrieved, themessage is no longer available in the queue.

SharedMemory − Communicationbetweentwoor moreprocesses is achievedthrougha sharedpieceof memory amongallprocesses. Thesharedmemoryneedstobeprotectedfromeachother bysynchronizingaccess toall the processes.

Semaphores −Semaphoresaremeantfor synchronizingaccesstomultipleprocesses. Whenoneprocesswantsto access the memory (for reading or writing), it needs to be locked (or protected) and released when the access is removed. This needs to be repeated by all the processes to secure data. Signals −Signal is a mechanism to communication between multiple processes by way of signaling. This means a source process will send a signal (recognized by number) and the destination process will handle it accordingly.

Note−Almost alltheprogramsinthis tutorialarebasedonsystemcallsunder LinuxOperating System(executed in Ubuntu).

PROGRAM:

(A)ECHOSERVERUSING PIPE

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include<sys/types.h>
#defineBUFFER_SIZE256

int main() {
intpipefd[2];
pid_t pid;
if(pipe(pipefd)==-1){
perror("pipe");
exit(EXIT_FAILURE);
}
pid=fork();
if(pid==-1){
perror("fork");
exit(EXIT_FAILURE);
} else if (pid == 0) {
close(pipefd[0]);//Closereadingend
charmessage[]="Hellofromchildprocess!";
write(pipefd[1], message, sizeof(message));
close(pipefd[1]);
exit(EXIT_SUCCESS);
}else{
close(pipefd[1]); // Close writing end
char buffer[BUFFER_SIZE];
read(pipefd[0],buffer,BUFFER_SIZE);
printf("Receivedmessagefromchildprocess:%s\n",buffer);

close(pipefd[0]);
}
return0;
}
```

(B)ECHO SERVER USINGMESSAGESSOURCECODE:

Program:

```
#include <sys/ipc.h>
#include <stdio.h>
#include <string.h>
#include<sys/msg.h>
#include <stdlib.h>
struct msgbuf {
long mtype;
charmtext[20];
};

intmain(){
structmsgbufsend,recv; int
qid, pid, len;

qid=msgget((key_t)0X2000,IPC_CREAT|0666); if
(qid == -1) {
 perror("\nMessagequeuecreationfailed");
 exit(1);
}

send.mtype=1;
strcpy(send.mtext,"\nhelloiamparent"); len
= strlen(send.mtext);

pid=fork();
if(pid>0){
if(msgsnd(qid,&send,len,0)==-1){
 perror("\n Message sending failed");
 exit(1);
 }
 printf("\nMessagehasbeenposted");
 sleep(2);
 if(msgrcv(qid,&recv,100,2,0)==-1){ perror("\n
 msgrcv error:");
 exit(1);
 }
 printf("\nMessagereceivedfromchild-%s\n",recv.mtext);
 }elseif(pid==0){
 if (msgrcv(qid, &recv, 100, 1, 0) == -1) {
 perror("\nChildmessagereceivedfailed");
 exit(1);
 }
 printf("\nReceivedfromparent-%s",recv.mtext); send.mtype
 = 2;
 strcpy(send.mtext,"\nhiiamchild"); len
 = strlen(send.mtext);
 if(msgsnd(qid,&send,    len,0)==    -1){
 perror("\nChildmessagesendfailed");
 exit(1);
 }
 }
return0;
  }
```

OUTPUT:

(A)

Client sending message: Hello from client!
Serverreceivedmessage:Hellofromclient!
Client receivedechoedmessage:Hellofromclient!

(B)

Messagehasbeenposted
Received from parent - hello i am parent
Messagereceived fromchild -hiiamchild

RESULT:

EX:6	**Implementation of Semaphores**
DATE:	

AIM:

ALGORITHM:

A semaphore is a counter used to synchronize access to a shared data amongst multiple processes.To obtain a shared resource, the process should: o Test the semaphore that controls the resource.OIfvalue ispositive, it gainsaccessand decrementsvalueofsemaphore.oIfvalue iszero, the process goes to sleep and awakes when value is > 0.

Whenaprocessrelinquishesresource,it incrementsthevalueofsemaphoreby1.

Producer-Consumer problem

Aproducerprocessproducesinformationtobeconsumed byaconsumerprocessAproducer canproduce one item while the consumer is consuming another one. With bounded-buffer size, consumer must wait if buffer is empty, whereas producer must wait if buffer is full. The buffer canbe implemented using any IPC facility.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#defineBUFFER_SIZE5

intbuffer[BUFFER_SIZE];
sem_t empty, full;
pthread_mutex_t mutex;int
in = 0, out = 0;

void*producer(void*arg){
int item;
for(inti=0;i<BUFFER_SIZE;i++) {
item=rand()%100;//Producearandomitem
sem_wait(&empty);
pthread_mutex_lock(&mutex);
buffer[in] = item;
printf("Produced: %d\n", item);
in= (in+ 1)%BUFFER_SIZE;
pthread_mutex_unlock(&mutex);
sem_post(&full);
sleep(1);//Sleepforsometimetosimulateproductiontime
}
return NULL;
}
void*consumer(void*arg){
int item;
for(int i=0;i<BUFFER_SIZE;i++){
sem_wait(&full);
pthread_mutex_lock(&mutex);
 item = buffer[out];
printf("Consumed: %d\n", item);
out =(out +1)% BUFFER_SIZE;
 pthread_mutex_unlock(&mutex);
sem_post(&empty);
sleep(2);
}


        return NULL;
```

```
}

intmain(){
pthread_tproducer_thread, consumer_thread;

sem_init(&empty,0, BUFFER_SIZE);
sem_init(&full, 0, 0);
pthread_mutex_init(&mutex,NULL);
pthread_create(&producer_thread, NULL, producer, NULL);
pthread_create(&consumer_thread,NULL,consumer,NULL);

pthread_join(producer_thread, NULL);
pthread_join(consumer_thread,NULL);
sem_destroy(&empty);
sem_destroy(&full);
pthread_mutex_destroy(&mutex);return
0;
}
```

OUTPUT:

```
Produced: 45
Produced: 72
Produced: 33
Produced: 89
Produced: 17
Consumed:45
Consumed:72
Consumed:33
Consumed:89
Consumed:17
```

RESULT:

EX:7	**Implementation Deadlock Detection Algorithm**
DATE:	

AIM:

ALGORITHM:

PROGRAM:

```
#include<stdio.h>

int main() {
int found, flag, l, i, j, k = 1, sum= 0, tp, tr;
intp[8][8],c[8][8],m[8],r[8],a[8],temp[8];

 printf("EnterNo.ofProcesses:");
 scanf("%d", &tp);
 printf("EnterNo.ofResources:");
 scanf("%d", &tr);

 printf("\nEnterClaim/Requestmatrix:\n"); for
 (i = 1; i <= tp; i++)
 for(j=1;j<=tr;j++)
 scanf("%d",&c[i][j]);

  printf("\nEnterAllocationmatrix:\n"); for
  (i = 1; i <= tp; i++)
  for(j=1;j<=tr;j++)
  scanf("%d",&p[i][j]);

  printf("\nEnterTotalresources:\n");
  for (i = 1; i <= tr; i++)
  scanf("%d",&r[i]);

 printf("\nEnterAvailabilityvector:\n"); for
  (i = 1; i <= tr; i++) {
  scanf("%d",&a[i]);
 temp[i] = a[i];
 }

  for(i=1;i<=tp;i++){ sum
  = 0;
  for(j=1;j<=tr;j++) sum
  += p[i][j];
  if(sum==0){
  m[k] = i;
  k++;
  }
```

```
    }
    for(i=1;i<=tp;i++){ for
    (l = 1; l <= tr; l++)
    if(i!=m[l]){
    flag = 1;
    for(j=1;j<=tr;j++){ if
    (c[i][j] < temp[j]) { flag
    = 0;
     break;
      }
      }
      if(flag==1){
      m[k] = i;
      k++;
      for(j=1;j<=tr;j++) temp[j]
      += p[i][j];
       }
       }
       }

    printf("Deadlockcausingprocessesare:"); for
    (j = 1; j <= tp; j++) {
    found=0;
    for(i=1;i<k;i++){ if (j
    == m[i]) {

    found=1;
    break;
     }
     }
if (found == 0)
printf("P%d",j);
    }

 return0;
     }
```

OUTPUT:

```
Enter No. of Processes :4
EnterNo.ofResources:3

EnterClaim/Requestmatrix: 0
0 0
200
302
211

EnterAllocationmatrix:
010
201
302
210

EnterTotalresources:
726

EnterAvailabilityvector:
313


Deadlockcausingprocessesare:P2P3P4
```

RESULT:

EX:8	**Bankers Algorithm forDeadLock Avoidance**
DATE:	

AIM:

ALGORITHM:

PROGRAM:

```
#include<stdio.h>
int main() {
 intoutput[10],ins[5],avail[5],allocated[10][5],need[10][5],max[10][5],p[10]; int k
 = 0, d = 0, t = 0, i, pno, j, nor, count = 0;

printf("Enternumberofresources:");
  scanf("%d", &nor);

 printf("\nEntermaxinstancesofeachresource\n"); for
(i = 0; i < nor; i++) {
avail[i]=0;
printf("%c=", (i+65));
 scanf("%d",&ins[i]);
  }

printf("\nEnterthenumberofprocesses:");
scanf("%d", &pno);

 printf("\nEnterAllocationmatrix\n");
for (i = 0; i < pno; i++) {
printf("Process %d: ", i);
 for (j = 0; j < nor; j++) {
scanf("%d",&allocated[i][j]);
avail[j] += allocated[i][j];
}
 }

printf("\nEnterMaxmatrix\n");
for (i = 0; i < pno; i++) {
printf("Process %d: ", i);
for (j = 0; j < nor; j++)
 scanf("%d", &max[i][j]);
}

//Calculatetheneedmatrix
for (i = 0; i < pno; i++) {
for (j = 0; j < nor; j++) {
need[i][j]=max[i][j]-allocated[i][j];
}
}

 printf("\nAvailableresourcesare:\n");
for (i = 0; i < nor; i++)
printf("%c=%d\n", (i+65), ins[i]- avail[i]);

 printf("\nNeedmatrixis:\n"); for
(i = 0; i < pno; i++) {
printf("Process %d: ", i);
for (j = 0; j < nor; j++)
printf("%d\t", need[i][j]);
printf("\n");
}
```

```
while(d!=-1){ d
 = -1;
  for(i=0;i<pno;i++){ count
 = 0;
 if(p[i] !=-1){
 for (j= 0;j< nor; j++){
  if(need[p[i]][j]<=avail[j])
  count++;
}
 if(count==nor){
output[k++]= p[i];
  for (j = 0; j < nor; j++)
 avail[j] += allocated[p[i]][j];
 p[i] = -1;
} else {
p[++d]=p[i];
 }
}
 }
}

 printf("\nProcessExecutionOrder:");
 printf("<");
 for (i = 0; i < k; i++)
printf("P%d",output[i]);
printf(">\n");

 return0;
}
```

OUTPUT:

```
Enternumberofresources:3

Entermaxinstancesofeachresource
A=7
B=2
C=6

EntertheNo.ofprocesses:5

EnterAllocationmatrix
Process 0: 0 1 0
Process 1:20 1
Process 2:30 2
Process 3:21 0
Process 4:00 2

EnterMaxmatrix
Process 0: 7 5 3
Process 1:32 2
Process 2:90 2
Process 3:22 2
Process 4:43 3



Availableresources are:
A=0
B=2
C=2

Needmatrix is:
 Process0:7    4   3
 Process1:1    2   1
 Process2:6    0   0
 Process3:0    1   2
 Process4:4    3   1

Process ExecutionOrder:< P1P3P0P2P4 >
```

RESULT:

EX:9	# Implementation of Thread in C
DATE:	

AIM:

ALGORITHM:

PROGRAM:

```
#include <stdio.h>
#include<pthread.h>
#defineNUM_THREADS5

void*threadFunction(void*threadID){
long tid;
tid= (long)threadID;
printf("HelloWorld!It'sme,thread#%ld!\n",tid); return
NULL;
}

intmain(){
pthread_tthreads[NUM_THREADS];
int rc;
longt;
for(t=0;t<NUM_THREADS; t++){
printf("Creatingthread%ld\n",t);
rc=pthread_create(&threads[t],NULL,threadFunction,(void*)t); if
(rc) {
printf("ERROR;returncodefrompthread_create() is%d\n",rc);
return -1;
}
}
for(t=0;t<NUM_THREADS; t++){
pthread_join(threads[t],NULL);
}
printf("Allthreadshavecompletedtheir execution. Exitingthemainthread.\n"); return 0;
}
```

OUTPUT:

```
Creatingthread0
HelloWorld!It'sme,thread#0! Creating
thread 1
HelloWorld!It'sme,thread#1! Creating
thread 2
HelloWorld!It'sme,thread#2! Creating
thread 3
HelloWorld!It'sme,thread#3! Creating
thread 4
HelloWorld!It'sme,thread#4!
Allthreadshavecompletedtheir execution.Exitingthemain thread.
```

RESULT:

EX:10	Implementation of Paging Technique of Memory Management
DATE:	

AIM:

ALGORITHM:

PROGRAM:

```
#include<stdio.h>

int main() {
 int ms,ps,nop,nop_remaining,np,rempages,i,j,x,y,offset,pa; int
s[10], fno[10][20];

printf("EnterPhysicalmemorysize:");
 scanf("%d", &ms);

 printf("EnterPagesize:");
scanf("%d", &ps);

 nop=ms/ps;
printf("\nNo. ofFramesavailableare:%d\n",nop);

printf("\nEnterno.ofprocesses:"); scanf("%d",
 &np);

nop_remaining = nop;
for(i=0;i<np;i++){
printf("\nEnterno.ofpagesforprocessP%d:", i+1);
scanf("%d", &s[i]);

if (s[i] > nop_remaining) {
printf("\nMemoryisFull");
break;
}

nop_remaining-=s[i];
printf("Enter PagetableforprocessP%d:", i+1);

for (j = 0; j < s[i]; j++)
scanf("%d", &fno[i][j]);
}

printf("\nEnterProcessNo.PageNo.andOffset:");
scanf("%d%d%d", &x, &y, &offset);

if(x<=0||x>np||y<0||y>=s[x]||offset <0||offset >=ps){ printf("\nInvalid
Process or Page No. or offset");
}else{
pa = fno[x - 1][y] * ps + offset;
printf("PhysicalAddressis:%d",pa);
}

return0;
}
```

OUTPUT:

```
EnterPhysicalmemorysize:1024 Enter
Page size : 256

No.ofFramesavailableare:4

Enter no. of processes : 2

Enter no. of pages for process P1 : 3
EnterPagetable forprocessP1:123

Enterno.ofpagesforprocessP2:2
EnterPagetableforprocessP2:45

EnterProcessNo.PageNo.andOffset :1250
Physical Address is : 562
```

RESULT:

EX:11	Implementation of Memory Allocation Methods for Fixed Partition
DATE:	

AIM:

ALGORITHM:

PROGRAM:

(A) FirstFitAllocation

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

#defineNUM_PARTITIONS5
boolpartitions[NUM_PARTITIONS]={false};
void allocateMemoryFirstFit(int processSize) {
int i;
for (i=0;i<NUM_PARTITIONS;++i){
 if(!partitions[i]&&i+1>=processSize){
 partitions[i] = true;
 printf("Memoryallocatedtoprocess ofsize%dat partition%d\n", processSize, i+1); return;
 }
 }
 printf("Memoryallocationfailedfor processofsize%d\n", processSize);
 }

intmain(){
int processes[]={2,4,3,5,1};
 intnumProcesses=sizeof(processes)/sizeof(processes[0]); for
 (int i = 0; i < numProcesses; ++i) {
 allocateMemoryFirstFit(processes[i]);
  }

 return0;
  }
```

(B) BestFitAllocation

Program:

```
#include <stdio.h>
#include<stdbool.h>
#defineNUM_PARTITIONS5
#defineMAX_PARTITION_SIZE100
boolpartitions[NUM_PARTITIONS]={false};
void allocateMemoryBestFit(int processSize) {
int bestFitPartition = -1;
 intminFreeSpace=MAX_PARTITION_SIZE+1;

 for(inti=0;i<NUM_PARTITIONS;++i){ if
 (!partitions[i] && i + 1 >= processSize) { int
 freeSpace = i + 1 - processSize;
 if(freeSpace<minFreeSpace){
 minFreeSpace = freeSpace;
 bestFitPartition = i;
  }
  }
  }

  if (bestFitPartition != -1) {
  partitions[bestFitPartition]=true;
  printf("Memoryallocatedtoprocessofsize%datpartition%d\n", processSize, bestFitPartition+ 1);
  }else{
  printf("Memoryallocationfailedforprocessofsize%d\n",processSize);
  }
}

intmain(){
int processes[]={2,4,3,5,1};
intnumProcesses=sizeof(processes)/sizeof(processes[0]); for
(int i = 0; i < numProcesses; ++i) {
allocateMemoryBestFit(processes[i]);
}
 return0;
}
```

(C) WorstFitAllocation

Program:

```
#include <stdio.h>
#include<stdbool.h>
#defineNUM_PARTITIONS5
#defineMAX_PARTITION_SIZE100
boolpartitions[NUM_PARTITIONS]={false};

voidallocateMemoryWorstFit(intprocessSize){
int worstFitPartition = -1;
intmaxFreeSpace=-1;

for(int i=0;i<NUM_PARTITIONS;++i){ if
(!partitions[i] && i + 1 >= processSize) { int
freeSpace = i + 1 - processSize;
if(freeSpace>maxFreeSpace){
maxFreeSpace = freeSpace;
worstFitPartition = i;
 }
}
}

 if (worstFitPartition != -1) {
 partitions[worstFitPartition]=true;
printf("Memoryallocatedtoprocessofsize%datpartition%d\n", processSize, worstFitPartition
+ 1);
}else{
printf("Memoryallocationfailedforprocessofsize%d\n",processSize);
 }
}

intmain(){
intprocesses[]={2,4,3,5,1};
intnumProcesses=sizeof(processes)/sizeof(processes[0]); for
(int i = 0; i < numProcesses; ++i) {
allocateMemoryWorstFit(processes[i]);
 }

 return0;
}
```

OUTPUT:

(A)

Memoryallocatedto processofsize2at partition1
Memoryallocatedto processofsize4at partition3
Memoryallocatedto processofsize3at partition2
Memoryallocatedto processofsize5at partition4
Memory allocation failed for process of size 1

(B)

Memoryallocatedto processofsize2at partition1
Memoryallocatedto processofsize4at partition3
Memoryallocatedto processofsize3at partition2
Memoryallocatedto processofsize5at partition4
Memory allocation failed for process of size 1

(C)

Memoryallocatedto processofsize1at partition5
Memoryallocatedto processofsize2at partition4
Memoryallocatedto processofsize3at partition3
Memoryallocatedto processofsize4at partition2
Memory allocation failed for process of size 5

RESULT:

EX:12	**Implementation of Page Replacement Algorithm**
DATE:	

AIM:

ALGORITHM:

(A) FIFOPageReplacement

PROGRAM:

```
#include<stdio.h>

int main() {
inti,j,l,rs[50],frame[10],nf,k,avail,count=0;

printf("Enterlengthofreferencestring:");
scanf("%d", &l);

 printf("Enterreferencestring:\n");
 for (i = 0; i < l; i++)
scanf("%d",&rs[i]);

printf("Enternumberofframes:"); scanf("%d",
&nf);

for(i=0;i<nf;i++) frame[i]
= -1;
j=0;

printf("\nRef.str\tPageframes\n");
for (i = 0; i < l; i++) {
printf("%4d\t\t", rs[i]);
avail= 0;

for(k=0;k<nf;k++){ if
(frame[k] == rs[i]) {
avail = 1;
break;
 }
}
if(avail==0){
frame[j]=rs[i];
j=(j+1)%nf;
count++;
for (k= 0; k< nf; k++)
printf("%d",frame[k]);
}
printf("\n");
}
printf("\nTotalnumber ofpagefaults:%d\n",count);

return0;
}
```

(B) LRUPageReplacement

PROGRAM:

```
#include<stdio.h>

intarrmin(int[],int);
int main() {
inti,j,len,rs[50],frame[10],nf,k,avail,count=0; int
access[10], freq = 0, dm;

 printf("LengthofReferencestring:");
scanf("%d", &len);

printf("Enterreferencestring:\n");
for (i = 1; i <= len; i++)
scanf("%d", &rs[i]);
printf("Enterno.offrames:");
scanf("%d", &nf);
 for(i=0;i<=len;i++)
frame[i] = -1;
j=0;
printf("\nRef.str\tPageframes");
for (i = 1; i <= len; i++) {
printf("\n%4d\t\t", rs[i]);
 avail= 0;
for(k=0;k<nf;k++){ if
(frame[k] == rs[i]) {
avail = 1;
 access[k]=++freq;
break;
}
}

if(avail==0){ dm
= 0;
 for(k=0;k<nf;k++){ if
 (frame[k] == -1) {
dm=1;
 break;
 }
}
 if (dm == 1) {
frame[k] = rs[i];
access[k]=++freq;
count++;
} else{
 j=arrmin(access,nf);
 frame[j] = rs[i];
 access[j] = ++freq;
 count++;
}

for (k = 0; k < nf; k++)
 printf("%d\t",frame[k]);
 }
 }
```

```
printf("\n\nTotalno.ofpagefaults:%d\n",count); return
 0;
}

int arrmin(int a[], int n) {
inti,min=a[0],index=0; for
(i = 1; i < n; i++) {
if(min>a[i]){
min = a[i];
index = i;
}
}
returnindex;
}
```

(C) LFUPageReplacement

PROGRAM:

```
#include<stdio.h>

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

printf("Enternoofpages:");
scanf("%d", &n);

printf("Enterthereferencestring:");
for (i = 0; i < n; i++)
scanf("%d",&p[i]);

printf("Enternoofframes:");
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>=0;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("\nThenoofpagefaultsis%d\n",c);
return 0;
}
```

Output:

(A)

Enterlengthofreferencestring:13 Enter
reference string:
7
0
1
2
0
3
0
4
2
3
0
3
2
Enternumberofframes:3

Ref.str	Page frames
7	7-1-1
0	70-1
1	701
2	201
	0
3	231
0	230
4	430
2	420
3	423
0	023
3	
2	

Totalnumberofpagefaults:10

(B)

LengthofReferencestring:12
Enter reference string :
1
2
3
4
5
1
2
3
4

5
6
7
Enterno.offrames:3

Ref.str	Page frames		
1	1	-1	-1
2	1	2	-1
3	1	2	3
4	4	2	3
5	4	5	3
1	4	5	1
2	4	2	1
3	3	2	1
4	4	2	1
5	4	5	1
6	6	5	1
7	6	7	1

Totalno.ofpagefaults:7

(C)

Enternoofpages:12
Enterthereferencestring:123451234567 Enter no
of frames: 3

1		
1	2	
1	2	3
4	2	3
4	5	3
4	5	1
6	5	1
6	7	1

Thenoofpagefaultsis7

RESULT:

EX:13	Implementation of Various File Organization Techniques
DATE:	

AIM:

ALGORITHM:

A)SingleLevelDirectory

PROGRAM:

```
#include<stdio.h>
int main() {
intnod;
printf("No.ofDirectories:");
scanf("%d", &nod);
printf("\nEnterthe directorydetails\n");

chardir[nod][20];
int nof[nod];
charfile[nod][20][20];

for (int i = 0; i < nod; i++) {
printf("\nDirectoryName:");
scanf("%19s",dir[i]);//Limit inputtopreventbufferoverflow printf("No. of
Files in the directory : ");
scanf("%d", &nof[i]);
printf("Enterthefilenames:\n");
for (int j = 0; j < nof[i]; j++)
scanf("%19s",file[i][j]);//Limitinputtopreventbufferoverflow
}

printf("\nDirectoryFilenames\n");
for (int i = 0; i < nod; i++) {
printf("%s\t", dir[i]);
for(int j=0;j<nof[i];j++)
printf("%s ", file[i][j]);
printf("\n");
}

return0;
}
```

(B) Two-LevelDirectory

Program:

```
#include<stdio.h>
struct st {
char uname[10];
chardname[10][10];
charfname[10][10][15];
 int ds;
intsds[10];
}dir[10];

intmain(){
inti, j,k, n;
printf("No.ofUsers:");
 scanf("%d",&n);

for (i = 0; i < n; i++) {
printf("\nUser-%dName:",i+1);
scanf("%s", dir[i].uname);
printf("No. of folders: ");
scanf("%d", &dir[i].ds);

for (j = 0; j < dir[i].ds; j++) {
printf("\nEnter folder name: ");
scanf("%s", dir[i].dname[j]);
printf("No. of files: ");
scanf("%d", &dir[i].sds[j]);
printf("Enter filenames:\n");
for(k=0;k<dir[i].sds[j];k++)
scanf("%s", dir[i].fname[j][k]);
 }
}

printf("\n\tTwo-LevelDirectoryStructure\n");
printf("\nUser\tFolders\tFiles\n\n");

for (i = 0; i < n; i++) {
printf("%s\t",dir[i].uname);

for(j=0; j<dir[i].ds; j++){
printf("\t%s\t",dir[i].dname[j]);

for (k = 0; k < dir[i].sds[j]; k++)
printf("%-15s",dir[i].fname[j][k]);

 printf("\n");
}
 printf("\n");
}

return0;
}
```

(C) HierarchicalDirectoryStructure

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include<graphics.h>

structtree_element{
char name[20];
intx,y,ftype,lx,rx,nc,level; struct
tree_element *link[5];
};

typedefstructtree_elementnode;
voidcreate(node**root,intlev,char*dname,intlx,intrx,intx){ int i,
 gap;
if(*root==NULL){
(*root) = (node *)malloc(sizeof(node));
printf("Enternameofdir/file(under%s):",dname);
fflush(stdin);
gets((*root)->name);
printf("Enter1forDir/2forfile:");
scanf("%d", &(*root)->ftype);
(*root)->level = lev;
(*root)->y=50+lev*50;
(*root)->x = x;
(*root)->lx=lx;
(*root)->rx=rx;
for (i = 0; i < 5; i++)
 (*root)->link[i]=NULL;
 if((*root)->ftype==1){
printf("Noofsubdirectories/files(for%s):",(*root)->name);
scanf("%d", &(*root)->nc);
 if((*root)->nc==0)
 gap = rx - lx;
 else
 gap = (rx - lx) / (*root)->nc;
 for(i=0;i<(*root)->nc;i++)
 create(&((*root)->link[i]), lev+1, (*root)->name, lx+gap*i, lx+gap*i+gap, lx+gap*i+ gap / 2);
  }else
  (*root)->nc=0;
}
}
voiddisplay(node*root){
int i;
settextstyle(2,0,4);
 settextjustify(1, 1);
setfillstyle(1,BLUE);
setcolor(14);
if(root!=NULL){
```

```
 for(i=0;i<root->nc;i++)
 line(root->x,root->y,root->link[i]->x,root->link[i]->y);
 if (root->ftype == 1)
  bar3d(root->x-20,root->y-10, root->x+20,root->y+10,0,0); else
fillellipse(root->x, root->y, 20, 20);
outtextxy(root->x,root->y,root->name);
for (i = 0; i < root->nc; i++)
display(root->link[i]);
}
}

intmain(){
 intgd=DETECT,gm; node
*root;
root=NULL; clrscr();
create(&root,0,"root",0,639,320); clrscr();
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
display(root);
getch();
closegraph();
return 0;
}
```

OUTPUT:

(A)

```
No. of Directories : 2

Enterthedirectorydetails

Directory Name : Directory1
No.ofFilesinthedirectory:3
Enter the filenames:
file1.txt
file2.txt
file3.txt

Directory Name : Directory2
No.ofFilesinthedirectory:2
Enter the filenames:
file4.txt
file5.txt

DirectoryFilenames
Directory1file1.txtfile2.txtfile3.txt
Directory2file4.txt file5.txt
```

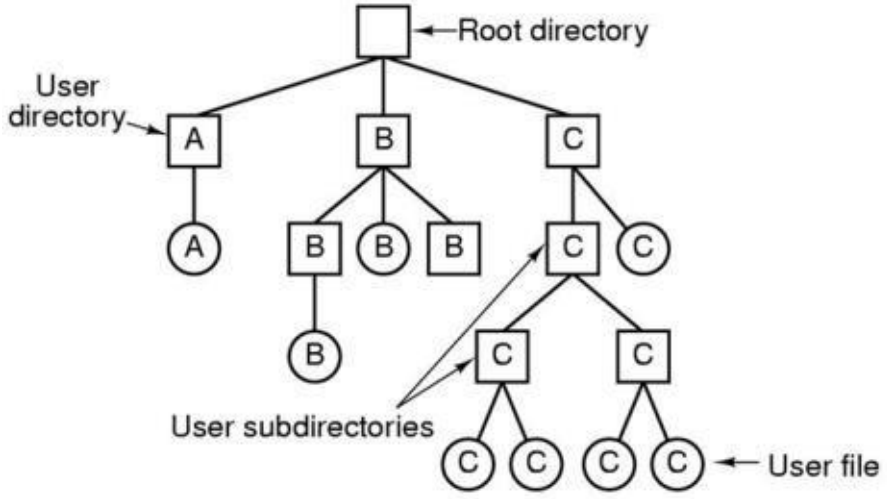
(B)

```
No.ofUsers:2

User-1Name:Alice
No. of folders: 2

Enterfoldername:Documents
No. of files: 3
Enterfilenames:
file1.txt
file2.txt
file3.txt

Enterfoldername:Pictures
No. of files: 2
Enterfilenames:
pic1.jpgpic2.jpg

User-2Name:Bob
No. of folders: 1

Enterfoldername:Videos
No. of files: 2
Enterfilenames:
video1.mp4
video2.mp4
```

Two-LevelDirectoryStructure

User	Folders	Files		
Alice	Documents	file1.txt	file2.txt	file3.txt
	Pictures	pic1.jpg	pic2.jpg	
Bob	Videos	video1.mp4	video2.mp4	

(C)



A hierarchical directory system.

RESULT:

EX:14	Implementation of FileAllocation Strategies
DATE:	

AIM:

ALGORITHM:

(A) ContiguousAllocation

Program:

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

intnum=0,length[10],start[10];
char fid[20][4], a[20][4];

voiddirectory(intst){
int i;
printf("\nFileStartLength\n");
for(i = st; i < num; i++)
printf("%-4s%3d%6d\n",fid[i],start[i],length[i]);
}

void display() {int
 i;
 printf("%4s","");
 for(i=0;i<20;i++) printf("%4d",
 i);
printf("\n");
 printf("%4s", "");
 for(i=0;i<20;i++)
 printf("%4s", a[i]);
 printf("\n");
}

intmain(){
inti,n,k,temp,st,nb,ch,flag; char
  id[4];
  for(i=0;i<20;i++)
  strcpy(a[i], "");
  printf("Diskspacebeforeallocation:\n");
  display();
   do{
   printf("\nEnterFilename(max3char):"); scanf("%s",
   id);
printf("Enterstartblock:");
   scanf("%d", &st);
printf("Enterno.ofblocks:");
   scanf("%d", &nb);
   strcpy(fid[num], id);
   length[num] = nb;
   flag= 0;
    if((st+nb)>20){
    printf("Requirementexceedsrange\n");
    continue;
    }
    for(i=st;i<st+nb;i++){
    if(strcmp(a[i], "") != 0)
    flag= 1;
    }
```

```
if(flag==1){
printf("Contiguousallocationnotpossible.\n");
continue;
}
start[num]= st;
for(i=st; i<(st+nb); i++)
strcpy(a[i], id);
printf("Allocation done\n");
num++;
printf("\nAnymoreallocation(1.yes/2.no)?:");
scanf("%d", &ch);
} while(ch == 1);
printf("\n\t\t\tContiguous Allocation\n");
printf("Directory:");
directory(0);
printf("\nDiskspaceafterallocation:\n");
display();
printf("\n");
return 0;
        }
```

(B) LinkedAllocation:

Program:

```
#include  <stdio.h>
#include<stdlib.h>
#include<string.h>

structblock{
  int bno;
structblock *next;
};

structfiletable{
char  name[20];
int nob;
structblock *sb;
};

int main() {
int i, j, n;
charstr[20];
structblock *temp;

printf("Enterno.offiles:");
scanf("%d", &n);

structfiletableft[n];//Definearrayoffiletable for

(i = 0; i < n; i++) {
printf("\nEnterfilename%d:",i+1);
scanf("%s", ft[i].name);
printf("Enter noofblocksinfile%d:", i+1); scanf("%d",
 &ft[i].nob);

 ft[i].sb=(structblock*)malloc(sizeof(structblock));
 temp = ft[i].sb;

 if (temp == NULL) {
printf("Memoryallocationfailed.\n");
return 1; // Exit program
}

 printf("Enterthediskblocks:");
scanf("%d", &temp->bno);
temp->next=NULL;

 for(j=1;j<ft[i].nob;j++){
temp->next=(structblock*)malloc(sizeof(structblock)); if
  (temp->next == NULL) {
printf("Memoryallocationfailed.\n");
return 1; // Exit program
 }
temp = temp->next;
scanf("%d",&temp->bno);
temp->next = NULL;
```

```
 }
 }

 printf("\nEnterfilenametobesearched:");
  scanf("%s", str);

  for (i= 0;i< n;i++){
 if(strcmp(str,ft[i].name)==0) break;
 }

  if(i== n)
 printf("\nFileNotFound");
 else {
 printf("\nFilename\tNo.ofBlocks\tBlocksOccupied\n");
 printf("%s\t\t%d\t", ft[i].name, ft[i].nob);
  temp=ft[i].sb;

  while (temp != NULL) {
printf("%d->",temp->bno);
temp = temp->next;
 }
 printf("NULL\n");
  }

 //Freeallocatedmemory for
 (i = 0; i < n; i++) { temp =
 ft[i].sb;
 structblock *next;
 while(temp!=NULL){
  next = temp->next;
  free(temp);
  temp=next;
 }
  }

  return0;
 }
```

(C) IndexedAllocation

Program:

```
#include<stdio.h>

int main() {
intn,m[20],i,j,sb[20],s[20],b[20][20],x;

 printf("Enterno.offiles:");
 scanf("%d", &n);

for (i= 0; i<n;i++){
 printf("Enter startingblockandsizeoffile%d:", i+1); scanf("%d
%d", &sb[i], &s[i]);
printf("Enternumberofblocks occupiedbyfile%d:",i+1);
 scanf("%d", &m[i]);
 printf("Enterblocksoffile%d:",i+1); for (j
= 0; j < m[i]; j++)
scanf("%d",&b[i][j]);
}

printf("\nFile\tIndex\tLength\n");
for (i = 0; i < n; i++) {
printf("%d\t%d\t%d\n",i+1,sb[i],s[i]);
}

 printf("\nEnterfileindextoviewdetails:");
 scanf("%d", &x);

 if (x <= n && x > 0) {
printf("Filename:%d\n",x); i
= x - 1;
 printf("Index:%d\n",sb[i]);
 printf("Size: %d\n", s[i]);
 printf("Blocks occupied:");
 for (j = 0; j < m[i]; j++)
 printf(" %d", b[i][j]);
} else{
printf("Invalidfileindex!");
}

 return0;
}
```

OUTPUT:

(A)

```
 Diskspacebeforeallocation:
012345678910111213141516171819

EnterFilename(max3char):A
Enter start block : 0
Enterno.ofblocks:4
Allocation done

Anymoreallocation(1.yes/2.no)?:1 Enter

File name (max 3 char) : B
Enter start block : 5
Enterno.ofblocks:3
Allocation done

Anymoreallocation(1.yes/2.no)?:1 Enter

File name (max 3 char) : C
Enter start block : 10
Enterno.ofblocks:5
Allocation done

Anymoreallocation(1.yes/2.no)?:2

Contiguous Allocation
Directory:
FileStartLength
A    0     4
B    5     3
C    10    5

Diskspaceafterallocation:
   012345678910111213141516171819
   AAAABBBCCCCC
```

(B)

```
Enter no. of files: 2

Enterfilename1:file1
Enternoofblocks infile1 :3
Enter thediskblocks :1 2 3

Enterfilename2:file2
Enter noofblocksinfile2:2
Enter the disk blocks : 4 5

Enter file name to be searched : file1

Filename    No.ofBlocksBlocksOccupied

file1           3           1 -> 2-> 3-> NULL
```

(C)

```
Enterno.offiles:2
Enter starting block and size of file 1: 1 3
Enternumberofblocksoccupiedbyfile1:3 Enter
blocks of file 1: 1 2 3
Enter starting block and size of file 2: 5 2
Enternumberofblocksoccupiedbyfile2:2 Enter
blocks of file 2: 5 6

FileIndexLength
1   1   3
2   5   2

Enterfileindextoviewdetails:1 File

name: 1
Index:1
Size:3
Blocksoccupied:123
```

RESULT:

EX:15	Implementation of Various Disk Scheduling Algorithm
DATE:	

AIM:

ALGORITHM:

(A) FCFS

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>//Includestdlib.hforabsfunction

intmain(){
intqueue[20],n,head,i,j,seek=0,max,diff; float
avg;

 printf("Enterthemaxrangeofdisk:");
scanf("%d", &max);

printf("Enterthesizeofqueuerequest:");
scanf("%d", &n);

printf("Enter thequeueofdiskpositionstoberead:\n"); for(i =
1; i <= n; i++)
scanf("%d",&queue[i]);

printf("Entertheinitialheadposition:");
scanf("%d", &head);

queue[0]=head;

for(j= 0;j<= n-1;j++) {
diff=abs(queue[j+1]-queue[j]); seek
+= diff;
printf("Diskheadmovesfrom%dto%dwithseek%d\n", queue[j], queue[j+1],diff);
}

printf("Totalseektimeis%d\n",seek); avg

 = seek / (float) n;
 printf("Averageseektimeis%f\n",avg);

return0;
}
```

(B) SSTF

Program:

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

intmain(){
intqueue[100],t[100],head,seek=0,n,i,j,temp; float
avg;

printf("***SSTFDiskSchedulingAlgorithm***\n");
printf("Enter the size of Queue\t");
scanf("%d",&n);

printf("EntertheQueue\t");
for (i = 0; i < n; i++)
scanf("%d", &queue[i]);

printf("Entertheinitialheadposition\t");
 scanf("%d", &head);

for (i= 0; i<n;i++){
t[i]=abs(head-queue[i]);
}

for  (i=0;i<n-1;i++){  for
(j=i+1;j<n;j++){ if (t[i] >
t[j]) {
temp=t[i];
t[i] = t[j];
t[j]=temp;

temp = queue[i];
queue[i]=queue[j];
queue[j] = temp;
 }
 }
 }

for (i = 0; i < n - 1; i++) {
seek+=abs(head-queue[i]);
head = queue[i];
 }

printf("\nTotalSeekTimeis%d\t",seek); avg
= seek / (float) n;
printf("\nAverageSeekTimeis%f\t",avg);

return0;
}
```

OUTPUT:

(A)

```
Enter the max range of disk:200
Enterthesizeofqueuerequest:5
Enterthequeueofdiskpositionstoberead: 50
120
30
150
90
Enter theinitialheadposition: 100
Disk head moves from 100 to 50 with seek 50
Disk head moves from 50 to 120 with seek 70
Disk head moves from 120 to 30 with seek 90
Diskheadmovesfrom30to150withseek120 Disk
head moves from 150 to 90 with seek 60 Total
seek time is 390
Averageseektimeis 78.000000
```

(B)

```
***SSTFDiskSchedulingAlgorithm***
Enter the size of Queue: 5
Enter theQueue: 40 20 60 10 50
Entertheinitialheadposition:30

TotalSeekTimeis100
AverageSeekTimeis20.000000
```

RESULT:

EX:16	InstallanyGuessOperatingSystem like Linux using VM Ware
DATE:	

AIM:

ALGORITHM:

StepstoinstallLinuxoperatingsystemusingVM(virtualMachine):

Step1:BackUpYourExistingData!

Thisishighlyrecommendedthat youshouldtakebackupofyourentiredata before start with the installation process.

Step2:ObtainingSystemInstallationMedia

Downloadlatest DesktopversionofUbuntufromthislink: http://www.ubuntu.com/download/desktop Booting

Step3: TheInstallationSystem

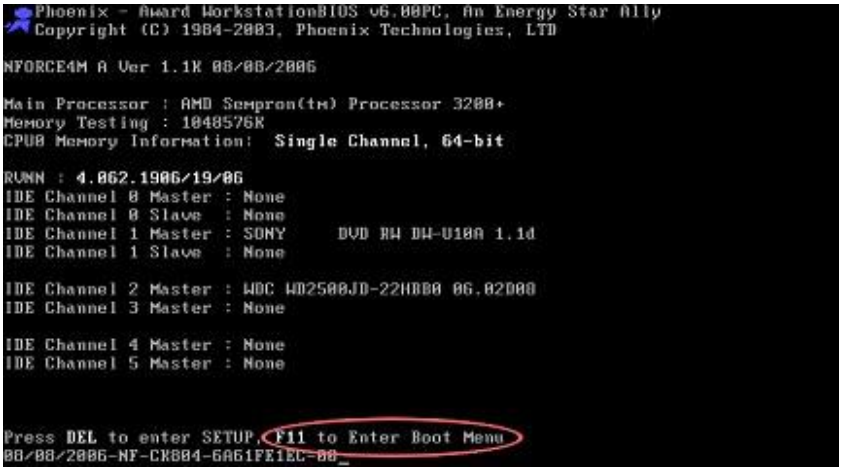
Thereareseveralwaystoboot theinstallationsystem. Someoftheverypopular waysare, Booting from a CD ROM, Booting from a USB memory stick, and Booting from TFTP.

HerewewilllearnhowtobootinstallationsystemusingaCDROM.

Beforebootingtheinstallationsystem, oneneedtochangetheboot order andset CD-ROMas first boot device.

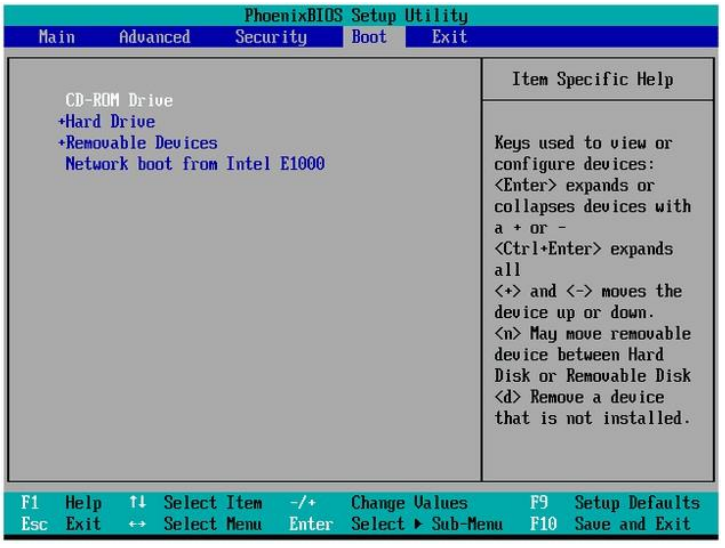
Step4: ChangingtheBootOrder ofaComputers

As your computer starts, press the DEL, ESC, F1, F2, F8 or F10 during the initial startup screen.DependingontheBIOS manufacturer,amenumayappear.However,consultthehardwaredocumentation for the exact key strokes. In my machine, its DEL key as shown in following screenshots



2. FindtheBootoptioninthesetuputility. Itslocationdepends onyour BIOS.SelecttheBootoption fromthe menu, you can now see the options Hard Drive, CD-ROM Drive, Removable Devices Disk etc..

3. Changetheboot sequencesettingsothat theCD-ROM is first.Seethelist of“ItemSpecificHelp”in right side of the window and find keys which is used to toggle to change the boot sequence.



4. InserttheUbuntuDiskinCD/DVD drive

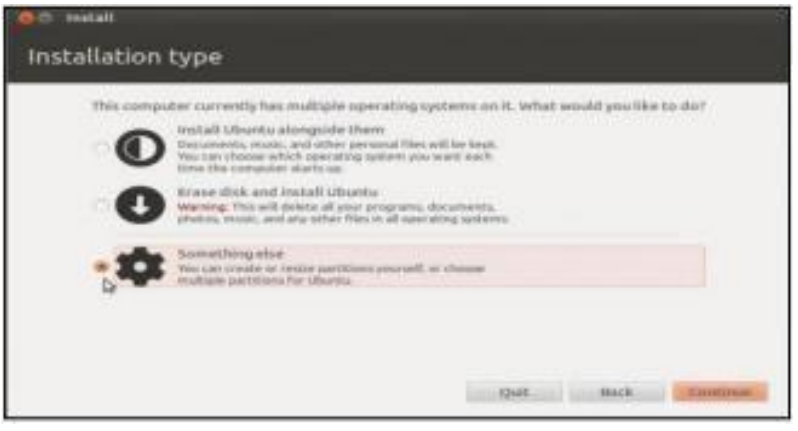
5. Saveyour changes. Instructionsonthescreentellyou howtosavethechanges onyour computer. The computer will restart with the changed settings.

MachineshouldbootfromCDROM,WaitfortheCDtoload

6. Inafewminutesinstallationwizardwillbestarted.Select yourlanguageandclick the"Install Ubuntu" button to continue...

8. Optionally,you can choose todownload updates while installing and/or Installthirdpartysoftware,suchasMP3support.Beaware,though,that ifyouselect Those options,the entire installation process will be longer!

9. Since we are going to createpartitions manually, select Something else, thenclick Continue. Keep in mind that evenifyou do not want to create partitions manually, It is better to select the same option as indicated here.This would insure that the installer will not over write your Windows, which will destroy your data. The assumptionhere isthat sdbwillbeused just for Ubuntu12.04, andthattheareno valuable data on it.



10. Whereareyou?

SelectyourlocationandClickthe"Continue"button.

Keyboardlayout

11. Selectyourkeyboardlayout andUK(English)andClickon“Continue”button.

12. Whoareyou?

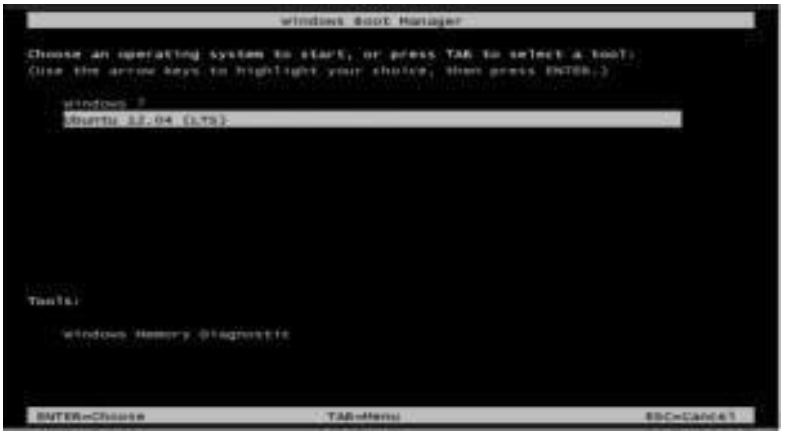
Fill in the fields with your real name, the name of the computer (automatically generated,but canbeoverwritten),username,and thepassword.Also atthisstep,there'san option called "Log in automatically."If you check it,you wil l automatically be logged into the Ubuntu desktop without giving the password.Option"Encrypt my homefolder,"will encryptyourhomefolder.Clickonthe

"Continue" button to continue...

13. NowUbuntu12.04LTS(PrecisePangolin)operatingsystemwillbeinstalled.

14. I will take approximately 10-12minutes(depending on computer's speed), apop-up windowwillappear,notifying youthattheinstallationiscomplete,and you'llneedto restart the computer in order to use Thenewyinstalled Ubuntuoperatingsystem.Clickthe"RestartNow"button

15. Please remove the CD and press the "Enter" keyto reboot. The computer will be restarted.Inafewseconds,youshouldseeWindows7′sbootmenuwithtwoentireslisted– Windows7andUbuntu12.04 (LTS). Then you may choose to boot into Windows 7 or Ubuntu 12.04 using the UP/Down arrow key.



16. Pleaseselect Ubuntu12.04(LTS)andpressEntertobootthe machine inUbuntu12.04 Linux.

17. Here youcanseetheusersonthemachine,Clickontheuser nameandenter the password and press Enter key to login.

18. WehavesuccessfullyinstallandlogintoUbuntu12.04LTS.

RESULT: