SPCA 206

MASTER OF COMPUTER APPLICATIONS

SECOND YEAR

THIRD SEMESTER

CORE PAPER - XVI

PRACTICAL - VI

OPERATING SYSTEMS AND COMPUTER GRAPHICS LAB

INSTITUTE OF DISTANCE EDUCATION

UNIVERSITY OF MADRAS

MASTER OF COMPUTER APPLICATIONS
SECOND YEAR - THIRD SEMESTER

CORE PAPER - XVI
PRACTICAL - VI : OPERATING SYSTEMS AND COMPUTER GRAPHICS LAB

WELCOME

Warm Greetings.

It is with a great pleasure to welcome you as a student of Institute of Distance Education, University of Madras. It is a proud moment for the Institute of Distance education as you are entering into a cafeteria system of learning process as envisaged by the University Grants Commission. Yes, we have framed and introduced Choice Based Credit System(CBCS) in Semester pattern from the academic year 2018-19. You are free to choose courses, as per the Regulations, to attain the target of total number of credits set for each course and also each degree programme. What is a credit? To earn one credit in a semester you have to spend 30 hours of learning process. Each course has a weightage in terms of credits. Credits are assigned by taking into account of its level of subject content. For instance, if one particular course or paper has 4 credits then you have to spend 120 hours of self-learning in a semester. You are advised to plan the strategy to devote hours of self-study in the learning process. You will be assessed periodically by means of tests, assignments and quizzes either in class room or laboratory or field work. In the case of PG (UG), Continuous Internal Assessment for 20(25) percentage and End Semester University Examination for 80 (75) percentage of the maximum score for a course / paper. The theory paper in the end semester examination will bring out your various skills: namely basic knowledge about subject, memory recall, application, analysis, comprehension and descriptive writing. We will always have in mind while training you in conducting experiments, analyzing the performance during laboratory work, and observing the outcomes to bring out the truth from the experiment, and we measure these skills in the end semester examination. You will be guided by well experienced faculty.

I invite you to join the CBCS in Semester System to gain rich knowledge leisurely at your will and wish. Choose the right courses at right times so as to erect your flag of success. We always encourage and enlighten to excel and empower. We are the cross bearers to make you a torch bearer to have a bright future.

With best wishes from mind and heart,

DIRECTOR

MASTER OF COMPUTER APPLICATIONS
SECOND YEAR - THIRD SEMESTER

CORE PAPER - XVI
PRACTICAL - VI : OPERATING SYSTEMS AND COMPUTER GRAPHICS LAB

COURSE WRITER

Dr. S. Sasikala
Assistant Professor in Computer Science
Institute of Distance Education
University of Madras
Chepauk, Chennnai - 600 005.

MASTER OF COMPUTER APPLICATIONS

SECOND YEAR

THIRD SEMESTER

Core Paper - XVI

PRACTICAL - VI

OPERATING SYSTEMS AND COMPUTER GRAPHICS LAB

SYLLABUS

Operating System Lab

1. Inter Process Communication (IPC) using Message Queues.
2. Implementations of wait and signal using counting semaphores.
3. Atomic Counter update problem.
4. Signaling processes.
5. Deadlock detection (for processes passing messages)
6. Process Scheduling: FCFS
7. Process Scheduling: Round Robin.
8. Two Process Mutual Exclusion.

Computer Graphics Lab:

1. Program to draw a line using DDA algorithm.
2. Program to draw a circle using Bresenham's algorithm.
3. Program to implement the Polygon clipping alogorithm.
4. Program to implement the Text clipping algorithm.
5. Program to implement the 2D Translation, 2D Rotation and 2D scaling.
6. Program to implement the 3D Translation, 3D Rotation and 3D scaling.
7. Program to implement the Shearing and Reflection of an object.

Cover page :1

INSTITUTE OF DISTANCE EDUCATION

RECORD OF PRACTICALS

M.C.A
(Secod Year)
2019-2020

Practical – VI
OPERATING SYSTEMS AND COMPUTER GRAPHICS LAB

Name	:
Enrolment Number	:
Group No	:

UNIVERSITY OF MADRAS
CHENNAI - 600 005

Cover page : 2

INSTITUTE OF DISTANCE EDUCATION
UNIVERSITY OF MADRAS
CHENNAI – 600 005

Certified that this is the Bonafide Record of work done by ______________________ With Enrolment Number____________________ of II year MCA Degree Course in the Institute of Distance Education, University of Madras during in the year _________respect of practical under Paper ______________________.

Date: **Co-ordinator**

Submitted for MCA Degree Course practical Examination held on _______________ at centre IDE, University of Madras.

Date: Examiners

(1) Name :

Signature:

(2) Name :

Signature:

OPERATING SYSTEM LAB

1. INTRODUCTION

History of Unix

Unix has a longer than any other popular operating system. Though many schools have contributed to its development, the initial contribution by The Bell Laboratory of AT&T and the University Of California, Berkley (UCB) are notable.

What is Unix?

Unix is an operating system. All computers have operating systems. An operating system is a software that act as an interface between the user and the computer hardware. An operating system acts a resources main memory, the hard disk, I/O devices and other peripherals.

The Shell prompt

Successful login into a Unix system is indicated by the appearance of a prompt called the shell prompt or system on the terminal. The character that appears as a prompt depends on the shell used.

Prompt	**Shell**
$ (dollar)	Bourne and korn shells(sh, bash, and ksh)
% (percent)	C shells (csh and tcsh)
# (hash)	Any shell as root

Commands in UNIX

UNIX has large number of commands. A list of some general features of a UNIX command is given below.

1. A UNIX command is a program written to perform certain specific action. All such programs have a name. for example, the program that is used to print today's date in a specific

manner has the name date and the program that is used to create a small file or display the contents of a file has the name cat and so on.

2. All unix commands are written using lower case letters. For example, cat, ls, who, date, and so on.
3. Almost all the Unix Command is cryptic. For example, cat stands for concatenation, ls stands for listing and so on. Unix commands were developed to be cryptic because it was developed by researchers for researchers and the early computer systems were very slow which demanded more time for typing, editing and executing long commands.
4. Unix Commands can have zero, one or more number of arguments associated with them.
5. UNIX commands can also have format specifies as well as options associated with them. Format specifies, whenever present, are indicated by the + character. Option, whenever present, are indicated by the (-). There could be many number of options associated with a command. It is interesting to note that the listing command (ls) has early nearly tow dozens options that could be used with it.
6. A current Unix command can be killed by using either <delete> or <ctrl-u> command.
7. Commands can be given to the system even when a command given earlier is being executed in the background. This is not possible with the Bourne shell, sh.

UNIX COMMANDS

File Commands	Description of Commands
1. ls	Directory listing
2. ls -al	Formatted listing with hidden files
3. ls -lt	Sorting the Formatted listing by time modification
4. cd dir	Change directory to dir
5. cd	Change to home directory
6. pwd	Show current working directory
7. mkdir	dir Creating a directory dir
8. cat >file	Places the standard input into the file
9. more file	Output the contents of the file
10. head file	Output the first 10 lines of the file
11. tail file	Output the last 10 lines of the file
12. tail -f	file Output the contents of file as it grows,starting with the last 10 lines
13. touch file	Create or update file
14. rm file	Deleting the file
15. rm -r	dir Deleting the directory
16. rm -f	file Force to remove the file
17. rm -rf	dir Force to remove the directory dir

18. cp file1 file2	Copy the contents of file1 to file2
19. cp -r dir1 dir2	Copy dir1 to dir2;create dir2 if not present
20. mv file1 file2	Rename or move file1 to file2,if file2 is an existing directory
21. In -s	file link Create symbolic link which link to file

Process management

1. ps	To display the currently working processes
2. top	Display all running process
3. kill pid	Kill the process with given pid
4. killall proc	Kill all the process named proc
5. pkill pattern	Will kill all processes matching the pattern
6. bg	List stopped or background jobs,resume a stopped job in the background
7. fg	Brings the most recent job to foreground
8. fg n	Brings job n to the foreground

File permission

1. chmod	octal file Change the permission of file to octal,which can be found separately for user,group,world by adding,

• 4-read(r)

• 2-write(w)

• 1-execute(x)

Searching

1. grep	pattern file Search for pattern in file
2. grep -r	pattern dir Search recursively for pattern in dir
3. command | grep	pattern Search pattern in the output of a command
4. locate file	Find all instances of file
5. find .	-name filename Searches in the current directory (represented by a period) and below it, for files and directories with names starting with filename
6. pgrep pattern	Searches for all the named processes , that matches with the pattern and, by default, returns their ID System Info
1. date	Show the current date and time
2. cal	Show this month's calender
3. uptime	Show current uptime
4. w	Display who is on line
5. whoami	Who you are logged in as
6. finger u	ser Display information about user
7. uname -a	Show kernel information
8. cat /proc/cpuinfo	Cpu information
9. cat proc/meminfo	Memory information
10. man	command Show the manual for command
11. df	Show the disk usage

12. du	Show directory space usage
13. free	Show memory and swap usage
14. whereis	app Show possible locations of app
15. which	app Show which applications will be run by default

Compression

1. tar cf file.tar	file Create tar named file.tar containing file
2. tar xf file.tar	Extract the files from file.tar
3. tar czf file.tar.gz	files Create a tar with Gzip compression
4. tar xzf file.tar.gz	Extract a tar using Gzip
5. tar cjf file.tar.bz2	Create tar with Bzip2 compression
6. tar xjf file.tar.bz2	Extract a tar using Bzip2
7. gzip file	Compresses file and renames it to file.gz
8. gzip -d file.gz	Decompresses file.gz back to file

Network

1. ping	host Ping host and output results
2. whois	domain Get whois information for domains
3. dig	domain Get DNS information for domain
4. dig -x	host Reverse lookup host
5. wget	file Download file
6. wget -c	file Continue a stopped download

Unix shortcuts Command

1. ctrl+c	Halts the current command
2. ctrl+z	Stops the current command, resume with fg in the foreground or bg in the background
3. ctrl+d	Logout the current session, si
4. ctrl+w	Erases one word in the current line
5. ctrl+u	Erases the whole line
6. ctrl+r	Type to bring up a recent command
7. !!	Repeats the last command
8. exit	Logout the current session

LAB EXERCISES

1. Inter Process Communication (IPC) using Message Queues.

MESSAGE SENDER

****

```
#include<stdio.h>

#include<sys/ipc.h>

#include<sys/types.h>

#include<sys/msg.h>

#include<string.h>

#include<sys/io.h>

#include<stdlib.h>

main(int arg,char *argv[])

{

int nod,i;

char buf[50];

nod =msgget((key_t)270,IPC_CREAT|0666);

if(nod<0)

{

milar to exit

printf("error");

exit(0);

}

for(i=1;i<arg;i++)
strcat(buf,argv[i]);
int len=strlen(buf);
int val=msgsnd(nod,buf,len,0);
if(val==0)
{       printf("Msg of size of %d has been sent",len);
}
else
{
printf("msg Sent Error");
}
exit(0);
}
```

MESSAGE RECIEVER

```
*****************
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
main(int argc,char *argv[])
{
char buf[50];
int mid=msgget((key_t)270,IPC_CREAT|0666);
if(mid<0)
printf("error in creation");
msgrcv(mid,buf,atoi(argv[1]),0,0);

printf("Obtained Msg is : %s\n",buf);

}
```

OUTPUT

```
*******

[c1@localhost ~]$ vi ex1a.c

[c1@localhost ~]$ cc ex1a.c

[c1@localhost ~]$ ./a.out hello

Msg of size of 5 has been sent[c1@localhost ~]$

[c1@localhost ~]$ vi ex1b.c

[c1@localhost ~]$ cc ex1b.c

[c1@localhost ~]$ ./a.out 5

Obtained Msg is : hello
```

2. Implementations of wait and signal using counting semaphores.

```
#include<stdio.h>

#include<sys/io.h>

#include<sys/ipc.h>

#include<sys/types.h>

#include<sys/sem.h>

main()

{

int pid,x,semid1;

struct sembuf sop;

semid1=semget((key_t)0*20,1,IPC_CREAT|0666);
```

```
semctl(semid1,0,GETVAL,1);

pid=fork();

if(pid==0)

{

printf("Value of semaphore is %d\n",x);

sop.sem_num=0;

sop.sem_flg=0;

sop.sem_op=0;

semop(semid1,&sop,1);

printf("\n Child in c section");

x=semctl(semid1,0,GETVAL,1);

printf("\n Value of Sema phore %",x);

}

else

{

sleep(5);

printf("\n Parent before c section");

sop.sem_num=0;

sop.sem_flg=0;

sop.sem_op=0;

semop(semid1,&sop,1);

printf("\nValues of semaphore %d",x);

sleep(13);

sop.sem_num=0;
sop.sem_flg=0;
sop.sem_op=0;
printf("\n Parent process out section\n");
x=semctl(semid1,0,GETVAL,1);
printf("\n Value of Semaphore %d",x);
}
}
```

OUTPUT

******

```
[a45@lab3 ~]$ vi semaphores.c
[a45@lab3 ~]$ cc semaphores.c
[a45@lab3 ~]$ ./a.out
Value of semaphore is 32767
 Child in c section
 Value of Sema phore 0
 Parent before c section
Values of semaphore 32767
 Parent process out section
 Value of Semaphore 0
```

3. ATOMIC COUNTER UPDATE PROBLEM

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
```

```
int count=0;

void increment(void)

{

count=count+1;

}

int getcount(void)

{

return count;

}

main()

{

int num =0;

int pid;

int p;

while(count<3)

{

pid=fork();

if(pid!=0)

{

sleep(5);

printf("\n This is the Parent");

increment();

p=getcount();

printf("_Counter Incremented :%d",p);
}
else
{
printf("\n This is the  Child");
increment();
p=getcount();
printf("_Counter Incremented: %d",p);
}
}
}
```

OUTPUT

****

```
[a45@lab3 ~]$ cc counter_updates.c
[a45@lab3 ~]$ ./a.out
This is the  Child_Counter Incremented: 1
This is the  Child_Counter Incremented: 2
This is the  Child_Counter Incremented: 3
This is the  Child_Counter Incremented: 1
This is the Parent_Counter Incremented: 1
This is the  Child_Counter Incremented: 2
This is the  Child_Counter Incremented: 2
This is the Parent_Counter Incremented: 3
This is the  Child_Counter Incremented: 3
This is the Parent_Counter Incremented: 2
This is the  Child_Counter Incremented: 3
This is the Parent_Counter Incremented: 1
This is the Parent_Counter Incremented: 2
This is the Parent_Counter Incremented: 3
This is the  Child_Counter Incremented: 2
This is the Parent_Counter Incremented: 3
This is the Parent_Counter Incremented: 2
This is the  Child_Counter Incremented: 3
This is the Parent_Counter Incremented: 2
This is the Parent_Counter Incremented: 3
```

4. SIGNALING PROCESSES.

```
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>
int count;
void alarm1()
{
printf("\n Alarm Ring");
alarm(2);
system("date");
if(++count<3)
return;
printf("\n Alarm switched off");
exit(0);
}
main()
{
printf("\n Signalling Process ");
printf("\n Setting Alarm");
count=0;
signal(SIGALRM,alarm1);
alarm(2);
pause();
while(1);
}
```

OUTPUT

******

```
[a45@lab3 ~]$ vi signalling_process.c
[a45@lab3 ~]$ cc signalling_process.c
[a45@lab3 ~]$ ./a.out
 Signalling Process
 Setting Alarm
Wed May 13 06:57:43 IST 2009
 Alarm Ring
Wed May 13 06:57:45 IST 2009
 Alarm Ring
Wed May 13 06:57:47 IST 2009
 Alarm Ring
 Alarm switched off
```

5. Deadlock detection

```
#include<stdio.h>

#include<string.h>

#define max 10

main()

{

int i , j=1,totRes,n=0,f=0;

int dead[10],need[10],pno[10],maxRes[10],first[10],ava[10];

printf("\n DEADLOCK DETECTION ");

printf("\n  ===================");

printf("\n Enter the Total Number of process==");

scanf("%d",&n);

printf("\n Enter the Total Number of Resources==");

scanf("%d",&totRes);

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

{

pno[i]=i;

printf("\n Enter the maximum No. of Resources for Process %d —>",i);

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

printf("\n Enter the First requirement Time of Process %d —>",i);

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

}

ava[0]=totRes;
```

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

{

if(first[i]<ava[i-1])

{

need[i]=maxRes[i]-first[i];

ava[i]=ava[i-1]-first[i];

}

else

{

need[i]=maxRes[i];

ava[i]=ava[i-1];

}

}

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

{

if(need[i]!=0&&(need[i]>ava[n]))

{

f=1;

dead[j]=pno[i];

j++;

}

}

printf("\nProcess\tMaximum\tFirst\tNeed\tAvail");

printf("\nNo\tResources\tRequires\tResource\tResource");
```

```
for(i=1;i<=n;i++)
{
printf("\n%d\t%d\t\t%d\t%d\t\t%d",pno[i],maxRes[i],first[i],need[i],
ava[i]);
}
if(f==0)
printf("\n\tDeadlock has not occured\n");
else
{
for(i=1;i<j;i++)
printf("\n\t DeadLock has occured due to that Process %d", dead[i]);
}
return 0;
}
```

OUTPUT

****

```
[a45@lab3 ~]$ vi d_d.c
[a45@lab3 ~]$ cc d_d.c
[a45@lab3 ~]$ ./a.out
 DEADLOCK DETECTION
  ====================
 Enter the Total Number of process==3
 Enter the Total Number of Resources==10
 Enter the maximum No. of Resources for Process 1 —>4
 Enter the First requirement Time of Process 1 —>1
 Enter the maximum No. of Resources for Process 2 —>4
 Enter the First requirement Time of Process 2 —>2
 Enter the maximum No. of Resources for Process 3 —>4
 Enter the First requirement Time of Process 3 —>2
Process     Maximum          First          Need           Avail
No           Resources       Requires       Resource       Resource
1           4                1              3              9
2           4                2              2              7
3           4                2              2              5
       Deadlock has not occured
```

6. Process Scheduling: FCFS

```
#include<stdio.h>
#include<math.h>
main()
{
int i,j,n;
int temp,no[20];
int bt[10],at[10],st[10],ft[10],wt[10],tt[10],ttt=0,twt=0,tet;
float awt=0,att=0;
printf("PROCESS  SCHEDULING(FCFS)\n");
printf("Enter the no of processors:");
scanf("%d",&n);
for(i=1;i<=n;i++)
```

```
{
printf("enter the process ID:");
scanf("%d",&no[i]);
printf("enter the burst time:");
scanf("%d",&bt[i]);
printf("Enter the no of arrival time");
scanf("%d",&at[i]);
}
for(i=1;i<=n;i++)
{
if(i==1)
st[i]=at[i];
else
st[i]=st[i-1]+bt[i-1];
ft[i]=st[i]+bt[i];
wt[i]=st[i]-at[i];
tt[i]=ft[i]-at[i];
ttt=ttt+tt[i];
twt=twt+wt[i];
}
tet=ft[n]-st[1];
awt=twt/n;
att=ttt/n;
printf("nPROCESS BRUST ARRIVAL START FINISH WAITING TURN AROUND");
printf("\n ID    TIME TIME TIME TIME TIME TIME\n");
for(i=1;i<=n;i++)
{
printf("\n %d\t%d\t%d\t%d\t%d\t%d\t%d\n",no[i],bt[i],at[i],st[i],ft[i],wt[i],tt[i]);
}
printf("\nTOTAL ELAPSE TIME:%d\n",tet);
printf("AVERGAE WAITING TIME:%f\n",awt);
printf("AVERAGE TURN AROUND TIME:%f\n",att);
}
```

OUTPUT

****

```
[a45@lab3 ~]$ vi fcfs_process.c
[a45@lab3 ~]$ cc fcfs_process.c
[a45@lab3 ~]$ ./a.out
PROCESS  SCHEDULING(FCFS)
Enter the no of processors:3
enter the  process ID:1
enter the burst time:24
Enter the no of arrival time:0
enter the process ID:2
enter the burst time:3
Enter the no of arrival time:0
enter the process ID:3
enter the burst time:3
```

```
Enter the no of arrival time:0

PROCESS BRUST ARRIVAL START FINISH WAITING      TURN AROUND

ID     TIME        TIME     TIME   TIME      TIME               TIME

 1      240         0          24         0          24

 2      3      0     24          27        24         27

3       3      0     27         30         27            30

TOTAL ELAPSE TIME:30

AVERGAE WAITING TIME:17.000000

AVERAGE TURN AROUND TIME:27.000000
```

7.Process Scheduling: Round Robin.

PROGRAM:

```
#include<stdio.h>

struct

{

intpno,btm,sbtm,wtm,lst;

}

proc[10];

int main()

{

inti,pp=-1,ts,flag,count,ptm=0,n,twt=0,tttm=0;

printf("\n Round Robin");

printf("\n Enter the no of process in the ready queue:");

scanf("%d",&n);

printf("\n Enter the timeslice:");
```

```
scanf("%d",&ts);
for(i=0;i<n;i++)
{
printf("\n\t Burst time %d process    ",i+1);
scanf("%d",&proc[i].btm);
proc[i].wtm=proc[i].lst=0;
proc[i].pno=i+1;
proc[i].sbtm=proc[i].btm;
}
printf("\n process synchronisation");
do
{
flag=0;
for(i=0;i<n;i++)
if((count=proc[i].btm)>0)
{
flag=1;
count=(count>ts)?ts:count;
printf("\n process: %d",proc[i].pno,ptm+=count);
proc[i].btm-=count;
if(pp!=i)
{
pp=i;
proc[i].wtm+=ptm-proc[i].lst-count;
proc[i].lst=ptm;
}
}
}
while(flag);
printf("\n\t Process no\t\tWaiting time\t\t Turn Around Time\n");
for(i=0;i<n;i++)
{
twt+=proc[i].wtm;
tttm+=proc[i].wtm+proc[i].sbtm;
printf("\n\t%d\t\t\t%d\n",proc[i].pno,proc[i].wtm,proc[i].wtm+proc[i].sbtm);
}
printf("\n Total wati time %d",twt);
printf("\n Average waiting time %f",(float)twt/n);
printf("\n Total turnaround time %d",tttm);
printf("\n Avgtrunaround time %f",(float)tttm/n);
}
```

<u>OUTPUT:</u>

```
 Round Robin
 Enter the no of process in the ready queue:3
 Enter the timeslice:5
        Burst time 1 process 25
         Burst time 2 process 5
        Burst time 3 process 5
processsynchronisation
process: 1
process: 2
process: 3
process: 1
process: 1
process: 1
process: 1
        Process no          Waiting time          Turn Around Time
             1                   10                      35
             2                    5                     10
             3                   10                      15
 Total wait time :25
 Average waiting time :8.333333
 Total turnaround time :60
Avgtrunaroundtime :20.000000
```

8. Two Process Mutual Exclusion.

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
main()
```

```
{
int semid;
struct sembuf sop;
semid=semget((key_t)270,1,IPC_CREAT|0666);
semctl(semid,0,SETVAL,1);
printf("\n******MUTUAL EXCLUSION******");
if(fork()==0)
{
printf("\nChild before calling semphore\n");
sop.sem_num=0;
sop.sem_flg=SEM_UNDO;
sop.sem_op=-1;
semop(semid,&sop,1);
printf("\nChild enteretd critical section\n");
sleep(1);
sop.sem_op=1;
semop(semid,&sop,1);
printf("\nChild cuming out of critical section\n");
}
else
{
printf("\nParent before calling semaphore\n");
sop.sem_num=0;
sop.sem_flg=SEM_UNDO;
sop.sem_op=-1;
semop(semid,&sop,1);
printf("\n Parent entered critical section\n");
sleep(1);
sop.sem_op=1;
semop(semid,&sop,1);
printf("\nParent coming out of critical section");
}
printf("\n\n");
}
```

OUTPUT

```
*******
[a45@lab3 ~]$ vi mut_ex.c
[a45@lab3 ~]$ cc mut_ex.c
[a45@lab3 ~]$ ./a.out
******MUTUAL EXCLUSION******
Parent before calling semaphore
 Parent entered critical section
******MUTUAL EXCLUSION******
Child before calling semphore
Parent coming out of critical section
Child enteretd critical section
[a45@lab3 ~]$
Child cuming out of critical section
```

COMPUTER GRAPHICS

LESSON - 1

INTRODUCTION

The Computer Graphics language makes it possible for you to control the shape, appearance, and motion of objects drawn using programmable graphics hardware. It marries programmatic control of these attributes with the incredible speed and capabilities of today's graphics processors. Never before have computer graphics practitioners, whether artists or programmers, had so much control over the real-time images they generate.

Cg stands for "C for graphics." The C programming language is a popular, general-purpose language invented in the 1970s. Because of its popularity and clean design, C provided the basis for several subsequent programming languages. For example, C++ and Java base their syntax and structure largely on C. The Computer graphics language bases itself on C as well. If you are familiar with C or one of the many languages derived from C, then Computer graphics will be easy to learn.

Graphics Basic-Function

First of all we have to call the initgraph function that will intialize the graphics mode on the computer. initgraph have the following prototype.

void initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver);

Initgraph initializes the graphics system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode. Initgraph also resets all graphics settings (color, palette, current position, viewport, etc.) to their defaults, then resets graphresult to 0.

*graphdriver Integer that specifies the graphics driver to be used. we can give graphdriver a value using a constant of the graphics_drivers enumeration type.

*graphmode Integer that specifies the initial graphics mode

(unless *graphdriver = DETECT).

If *graphdriver = DETECT, initgraph sets *graphmode to the highest resolution available for the detected driver. *graphmode a value using a constant of the graphics_modes enumeration type.

pathtodriver Specifies the directory path where initgraph looks for graphics drivers (.BGI) first.

Functions

Cleardevice()

Clears all previous graphical outputs generated by the previous programs.Its a good practice to include this method at the starting of each program.

Syntax: cleardevice()

gotoxy()

This will initialize the graphics cursor to the specified co-ordiante.In C gotoxy function is used very frequently to locate the cursor at different locations whenever as necessary.

Syntax : gotoxy(x,y)

getpixel () -

Get the color of a specified pixel .

Syntax: getpixel(x,y)

putpixel () It will color the pixel specified by the co-ordinates.

Syntax: putpixel(x,y,color)

outtextxy()

This method is used to display a text in any position on the screen.The numeric coordinates are substituted for x and y.

Syntax: outtextxy(x,y,"HELLO")

rectangle()

Draws a rectangle according to the given parameter x and y are the top-left corner co-ordinates.

Syntax : rectangle(int left, int top, int right, int bottom)

circle() Draws a circle with x,y as the center .

Syntax: circle(x,y,radius)

line() Draws a line as per the given co-ordinates.

Syntax : line(int startx, int starty, int endx, int endy)

moveto()

Cursor is moved from the current location to the specified location dx,dy.These

parameters can also be used as incremental values.

Syntax : moveto(dx,dy)

lineto() Draws a line from its current location to the co-ordinate(x,y)

Syntax : lineto(x,y)

ellipse() Draws the ellipse with the specified angles and coordinates.

Syntax :

ellipse(x-centre,y-center,starting_angle,ending_angle,x_radius,y_radius)

drawpoly() Draws a polygon with (num_of_points +1) edges.The array 'points' int points[ ]=(x1,y1,x2,y2,x3,y3...)

Syntax : drawpoly(num_of_points + 1, points)

settextstyle() The fonts available are :TRIPLEX_FONT, SMALL_FONT

SANS_SERIE_FONT, GOTHIC_FONT

The direction can be changed as HORIZ_DIR or VERT_DIR,

The charecter size increases from 1 to 10

Syntax : settextstyle(DEFAULT_FONT,HORIZ_DIR,1)

setfillstyle()

The fill styles avaliable are SOLID_FILL, LINE_FILL, HATCH_FILL,

SLASH_FILL etc.

Syntax : setfillstyle(SOLID_FILL,RED)

setcolor() Sets the color

Syntax : setcolor(color_name)

delay() Cause a pause in execution of the program 1000ms= 1 second

Syntax : delay(100)

closegraph()

Terminates all graphics operations and revert the hardware back to the normal mode.

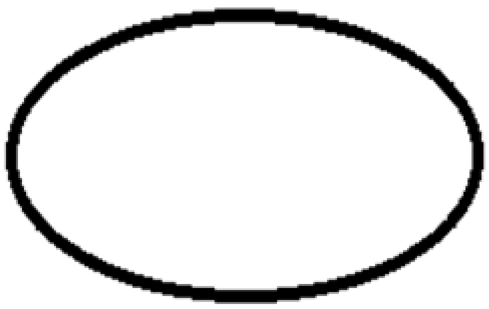
1.1 Write a C program to draw a circle

```
/* simple.c */

#include<graphics.h>

#include<conio.h>
```

```
void main()
{
    int gd=DETECT, gm;
    initgraph(&gd, &gm, "s:\\cpp\\bgi " );
    circle(100,100,50);
    getch();
    closegraph();
}
```

To run this program, you need **graphics.h** header file, **graphics.lib** library file and Graphics driver (BGI file) in the program folder. These files are part of Turbo C package. In all the programs listed here we use 640x480 VGA resolution. You need to make necessary changes to your programs according to your screen resolution. For VGA monitor, graphics driver used is EGAVGA.BGI.

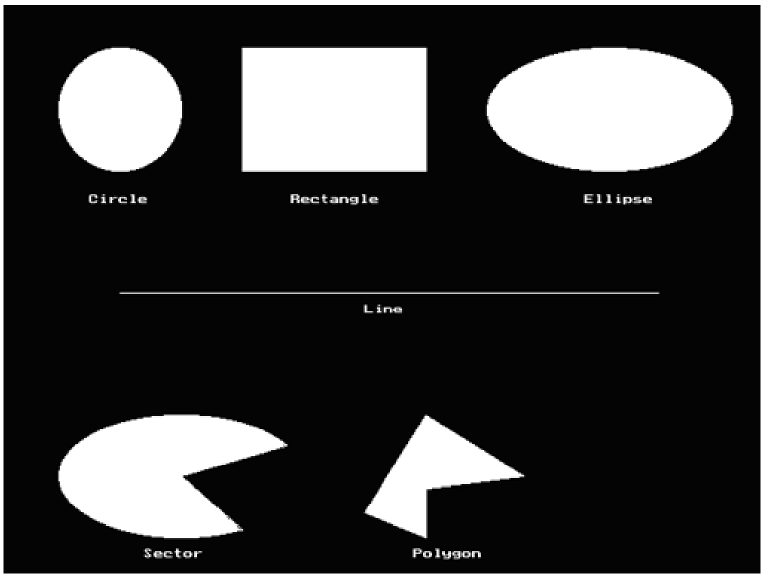
Output



1.2 Write a program to draw different shapes

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT, gm;
    int poly[12]={350,450, 350,410, 430,400, 350,350, 300,430, 350,450 };
    initgraph(&gd, &gm, “s:\\cpp\\bgi”);
    circle(100,100,50);
    outtextxy(75,170, “Circle”);
    rectangle(200,50,350,150);
    outtextxy(240, 170, “Rectangle”);
    ellipse(500, 100,0,360, 100,50);
    outtextxy(480, 170, “Ellipse”);
    line(100,250,540,250);
    outtextxy(300,260,”Line”);
    sector(150, 400, 30, 300, 100,50);
    outtextxy(120, 460, “Sector”);
    drawpoly(6, poly);
    outtextxy(340, 460, “Polygon”);
    getch();
    closegraph();
}
```

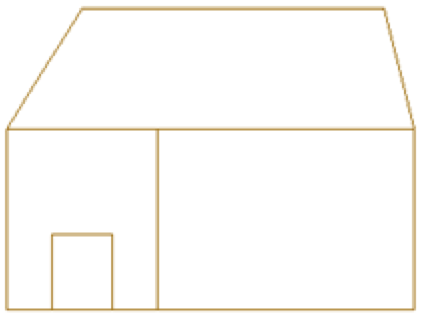
Output



1.3 Write a C program to draw Hut

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"s:\\cpp\\bgi");
setcolor(6);
rectangle(50,180,150,300);
rectangle(150,180,320,300);
rectangle(80,250,120,300);
line(100,100,50,180);
line(100,100,300,100);

line(100,100,300,100);

line(300,100,320,180);

getch();

closegraph();

}
```

Output



LESSON - 2

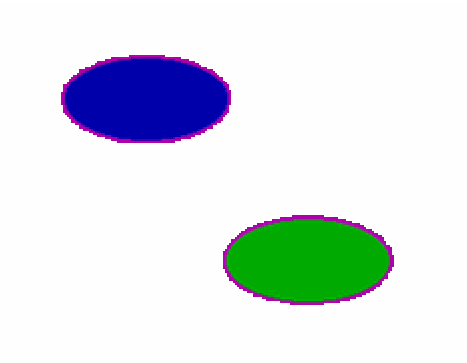
2.1 WRITE A C PROGRAM FOR FLOOD FILL ALGORITHM

Source code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void floodFill(int x,int y,int ncolor,int ocolor)
{
if(getpixel(x,y)==ocolor)
{
putpixel(x,y,ncolor);
floodFill(x+1,y,ncolor,ocolor);
floodFill(x-1,y,ncolor,ocolor);
floodFill(x,y+1,ncolor,ocolor);
floodFill(x,y-1,ncolor,ocolor);
}
delay(1);
}
void main()
{
int x,y,ncolor=BLUE,ocolor=WHITE;
int midx,midy;
int gd=DETECT,gm;
initgraph(&gd,&gm,”s:\\cpp\\bgi”);
cleardevice();
printf("Enter seed point:");
scanf("%d%d",&x,&y);
midx=getmaxx()/2;
midy=getmaxy()/2;
setbkcolor(5);
setpalette(ocolor,GREEN);
fillellipse(midx,midy,50,25);
fillellipse(midx+100,midy+100,50,25);
floodFill(x,y,ncolor,ocolor);
getch();
closegraph();
}
```

Output

```
Enter seed point:
340

240
```

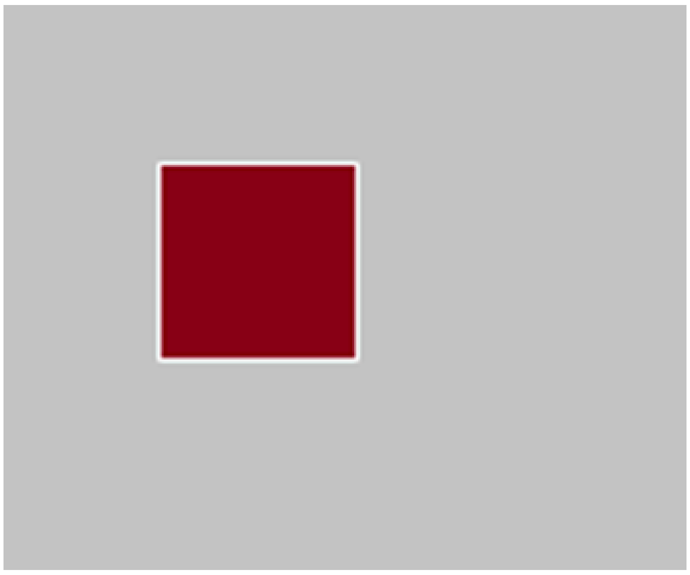


2.2 Write a C program for Boundary fill Algorithm

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void boundaryFill(int x,int y)

{

int interiorColor;

interiorColor=getpixel(x,y);

if((interiorColor !=WHITE)&&(interiorColor !=RED))

{

putpixel(x,y,RED);

boundaryFill(x+1,y);

boundaryFill(x,y+1);

boundaryFill(x-1,y);

boundaryFill(x,y-1);

}

delay(1);

}

void main()

{

void boundaryFill(int,int);

int x,y,n,i;

int gd=DETECT,gm;
```

```
        clrscr();
        initgraph(&gd,&gm,"s:\\cpp\\BGI");
        line(50,50,100,50);
        line(100,50,100,100);
        line(100,100,50,100);
        line(50,100,50,50);
        x=78,y=78;
        boundaryFill(x,y);
        getch();
        closegraph();
        }
```

Output

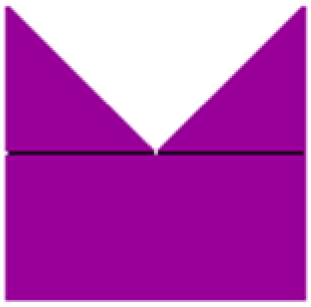


2.3 Write a C PROGRAM TO FILL POLYGON

```
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<process.h>
void main()
{
int graphdriver=DETECT,graphmode;
int i,j,d,p=1,x;
int a[12]={100,100,150,150,200,100,200,200,100,200,100,100};
initgraph(&graphdriver,&graphmode,"s:\\cpp\\bgi");
drawpoly(6,a);
for( i=100;i<200;i++)
        {    p=1;
for(j=100;j<=200;j++)
        {
         x=getpixel(j,i);
         for(d=0;d<11;d++)
         {
             if(j==a[d]&&i==a[d+1] )
             break;
             else
             {
              if(x>0&&d==10)
                        p++;
```

```
            if(p%2==0)
               putpixel(j,i,5);
            }}}}
      getch();
      closegraph();
      }
```

Output

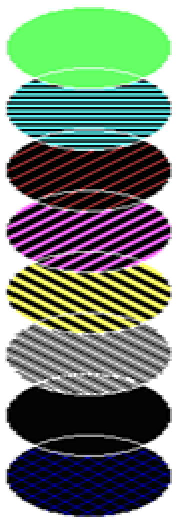


2.4 Write a program to implement the style option

```
      #include<stdio.h>
      #include<conio.h>
      #include<math.h>
      #include<dos.h>
      #include<graphics.h>
      void main()
      {
      int gm ,gr;
      int i, cx=40, cy=120;
      clrscr();
      detectgraph(&gm,&gr);
```

```
initgraph(&gm,&gr,”s:\\cpp\\BGI”);
rectangle(10,50,240,450);
cx=120;
cy=90;
for(i=1;i<=8;i++)
{
circle(cx,cy,30);
setfillstyle(i,9+i);
floodfill(cx,cy,15);
cy=cy+45;
}
outtextxy(300,150,”FILLSTYLE METHOD”);
getch();
}
```

Output



LESSON - 3

DIGITAL DIFFERENTIAL ANALYSER : (DDA ALGORITHM)

Algorithm

Step 1 : Get the endpoints of a line from the user.

(Xstart, Ystart) and (Xend, Yend) are the end points of a line.

Step 2 : Calculate dx and dy.

dx = Xend - Xstart

dy = Yend - Ystart

Step 3 : Calculate the slope 'm'

m = dx / dy;

Step 4 : If slope is less than or equal to 1(| m | <= 1), then the X and Y successors are calculated as follows.

Xk + 1 = Xk + 1/m

Yk + 1 = Yk + m

k starts from 1 and get incremented by 1 until endpoint(Xend) is reached.

Step 5 : If slope is greater than 1(| m | > 1), then the X and Y successors are calculated as follows.

Yk + 1 = Yk + 1

Xk + 1 = Xk + (1 / m)

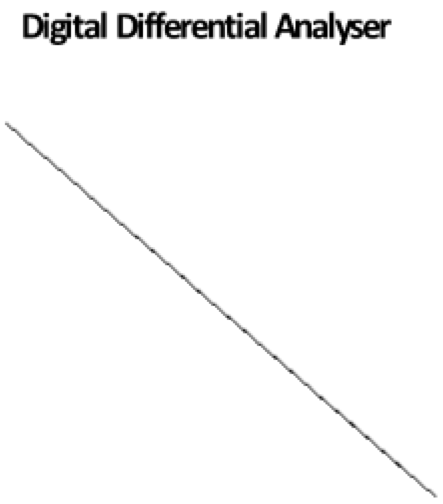
k starts from 1 and get incremented by 1 until endpoint(Yend) is reached.

1. Program to draw a line using DDA algorithm.

```
#include<graphics.h>

#include<conio.h>

#include<math.h>

void main()

{

int gdriver=DETECT,gmode, errorcode;

int x1,y1,x2,y2,k=0;

float dx,dy,steps,xinc,yinc,x,y,px,py;

initgraph(&gdriver,&gmode,”s:\\cpp\\bgi”);

printf(“Enter x1,y1:”);

scanf(“%d%d”,&x1,&y1);

printf(“Enter x2,y2:”);

scanf(“%d%d”,&x2,&y2);

dx=x2-x1;

dy=y2-y1;

steps=dx>dy?abs(dx):abs(dy);

xinc=dx/steps;

yinc=dy/steps;

clrscr();

x=x1;

y=y1;

for(;k<=steps;k++)

{
```

```
x+=xinc;
y+=yinc;
px=(int)x;
py=(int)y;
putpixel(px,py,0);
}
getch();
closegraph();
}
```

OUTPUT:



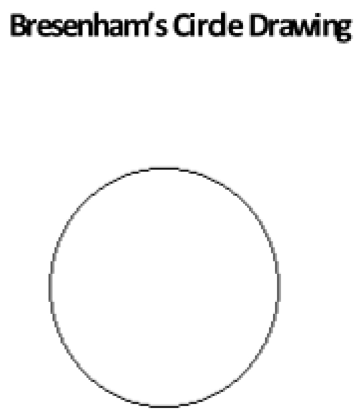
Digital Differential Analyser

2. Program to draw a circle using Bresenham's algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,xc,yc,r,x,y,Pk;
```

```
clrscr();
initgraph(&gd,&gm,"S:\\CPP\\BGI");
printf("*** Bresenham's Circle Drawing Algorithm ***\n");
printf("Enter the Value of X c\t");
scanf("%d",&xc);
printf("Enter the value of c c\t");
scanf("%d",&yc);
printf("Enter the Radius of Circle\t");
scanf("%d",&r);
x=0;
y=r;
putpixel(xc+y,yc-y,1);
Pk=3-(2*r);
for(x=0;x<=y;x++)
{
if(Pk<0)
{
y=y;
Pk=(Pk+(4*x)+6);
}
else
{
y=y-1;
Pk=Pk+((4*(x-y)+10));
}
putpixel(xc+x,yc-y,7);
putpixel(xc-x,yc-y,7);
putpixel(xc+x,yc+y,7);
putpixel(xc-x,yc+y,7);
putpixel(xc+y,yc-x,7);
putpixel(xc-y,yc-x,7);
putpixel(xc+y,yc+x,7);
putpixel(xc-y,yc+x,7);
delay(100);
}
getch();
}
```

OUTPUT:



3.Program to implement the Polygon clipping alogorithm.

```
POLYGON CLIPPING:

********************

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

#include <process.h>

#define TRUE 1

#define FALSE 0

typedef unsigned intoutcode;

outcodeCompOutCode(float x,float y);

enum  {  TOP = 0x1,

BOTTOM = 0x2,

RIGHT = 0x4,

LEFT = 0x8

};

floatxmin,xmax,ymin,ymax;

void clip(float x0,float y0,float x1,float y1)

{

outcode outcode0,outcode1,outcodeOut;

int accept = FALSE,done = FALSE;

outcode0 = CompOutCode(x0,y0);

outcode1 = CompOutCode(x1,y1);
```

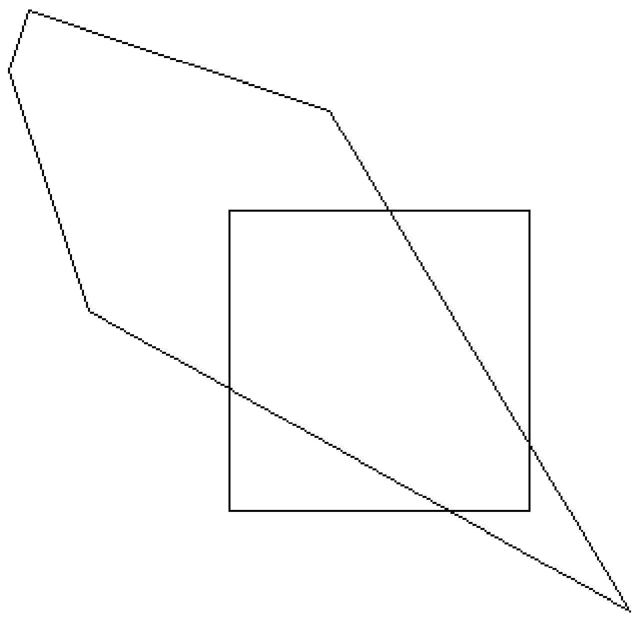
```
do
{
if(!(outcode0|outcode1))
{
accept = TRUE;
done = TRUE;
}
else
if(outcode0 & outcode1)
done = TRUE;
else
{
floatx,y;

outcodeOut = outcode0?outcode0:outcode1;
if(outcodeOut& TOP)
{
x = x0+(x1-x0)*(ymax-y0)/(y1-y0);
y = ymax;
}
else
if(outcodeOut& BOTTOM)
{
x = x0+(x1-x0)*(ymin-y0)/(y1-y0);
```

```
y = ymin;
}
else
if(outcodeOut& RIGHT)
{
y = y0+(y1-y0)*(xmax-x0)/(x1-x0);
x = xmax;
}
else
{
y = y0+(y1-y0)*(xmin-x0)/(x1-x0);
x = xmin;
}
if(outcodeOut==outcode0)
{
x0 = x;
y0 = y;
outcode0 = CompOutCode(x0,y0);
}
else
{
x1 = x;
y1 = y;
outcode1 = CompOutCode(x1,y1);
```

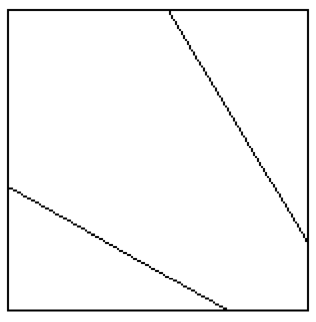
```
}
}
}while(done==FALSE);
if(accept)
line(x0,y0,x1,y1);
outtextxy(150,20,"POLYGON AFTER CLIPPING");

rectangle(xmin,ymin,xmax,ymax);
}
outcodeCompOutCode(float x,float y)
{
outcode code = 0;
if(y>ymax)
code|=TOP;
else
if(y<ymin)
code|=BOTTOM;
if(x>xmax)
code|=RIGHT;
else
if(x<xmin)
code|=LEFT;
return code;
}
```

```
void main( )

{

float x1,y1,x2,y2;

/* request auto detection */

intgdriver = DETECT, gmode, n,poly[14],i;

clrscr(  );

printf("Enter the no of sides of polygon:");

scanf("%d",&n);

printf("\nEnter the coordinates of polygon\n");

for(i=0;i<2*n;i++)

{

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

}

poly[2*n]=poly[0];

poly[2*n+1]=poly[1];

printf("Enter the rectangular coordinates of clipping window\n");

scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);

/* initialize graphics and local variables */

initgraph(&gdriver, &gmode, "s:\\cpp\\bgi");


outtextxy(150,20,"POLYGON BEFORE CLIPPING");

drawpoly(n+1,poly);

rectangle(xmin,ymin,xmax,ymax);

getch( );
cleardevice( );
for(i=0;i<n;i++)
clip(poly[2*i],poly[(2*i)+1],poly[(2*i)+2],poly[(2*i)+3]);
getch( );
restorecrtmode( );
}

/*
OUTPUT:
Enter the no of sides of polygon:5
Enter the coordinates of polygon
50
50
200
100
350
350
80
200
40
80
Enter the rectangular coordinates of clipping window
150
150
300
300*/
```

POLYGON BEFORE CLIPPING



POLYGON AFTER CLIPPING

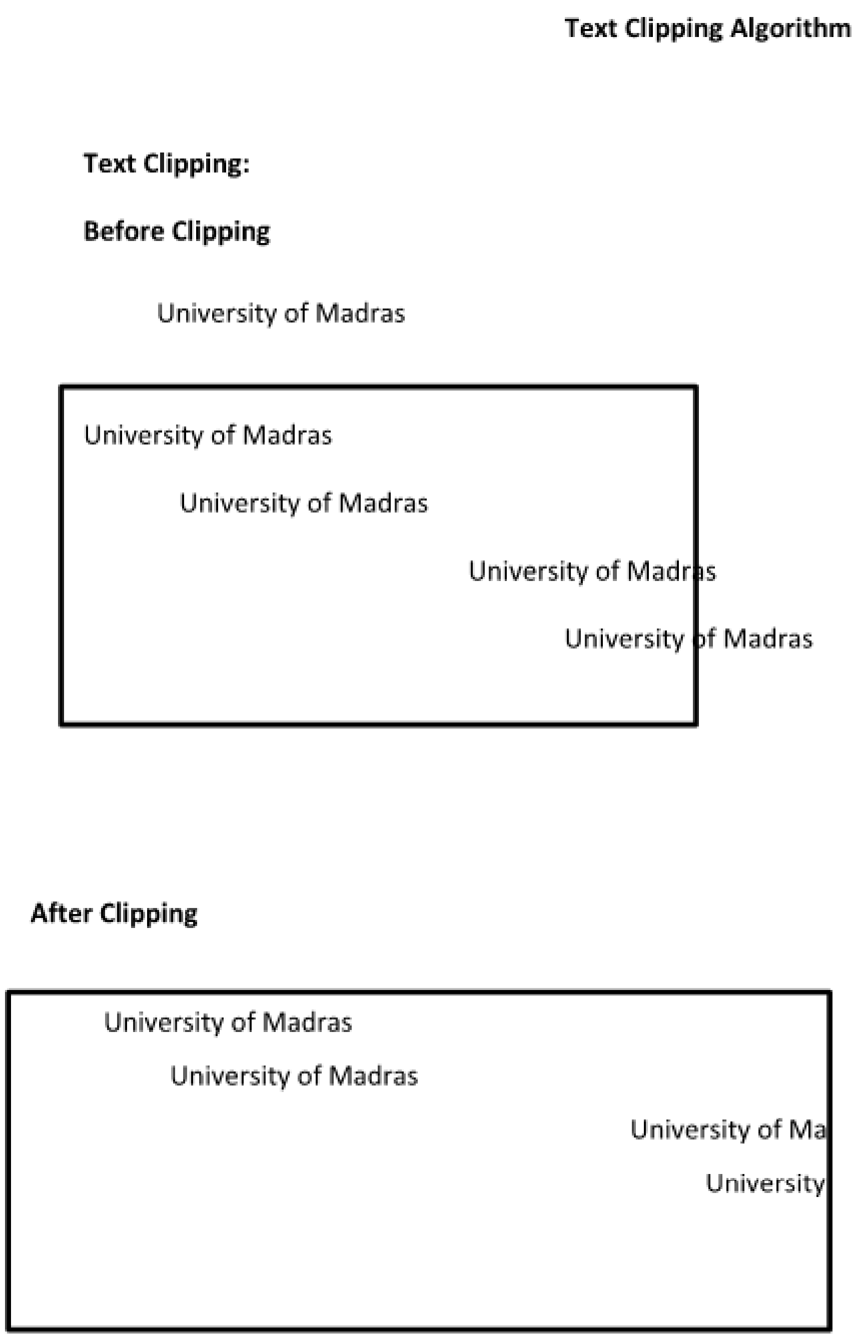


OUTPUT:

4. Program to implement the Text clipping algorithm.

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int xmin=70,ymin=150,ymax=350,xmax=550;

int xpos[6]={100,200,340,400,150,100};

int ypos[6]={98,250,193,220,550,150};

char s[25]=”University of Madras”;

void textclip1()

{

int i=0,j=0;

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

{

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

{

if(getpixel(i,j)==15)

{

if((i<=xmax)&&(i>=xmin)&&(j<=ymax)&&(j>=ymin))

putpixel(i,j,RED);

else

putpixel(i,j,0);

}
}
}
}
void main()
{
int gdriver=DETECT,gmode;
int i,opt;
initgraph(&gdriver,&gmode,"s:\\cpp\\bgi");
clearviewport();
printf("\n\t\t Text Clipping:");
rectangle(xmin,ymin,xmax,ymax);
settextstyle(8,0,3);
for(i=0;i<6;i++)
{
outtextxy(xpos[i],ypos[i],s);
}
printf("\n\t\t before Clipping");
rectangle(xmin,ymin,xmax,ymax);
getch();
printf("\n\t\t After Clipping");
textclip1();
getch();
closegraph();
}
```

OUTPUT:



```
                              Text Clipping Algorithm

Text Clipping:

Before Clipping

    University of Madras

University of Madras
     University of Madras
                    University of Madras
                          University of Madras

After Clipping

     University of Madras
        University of Madras
                              University of Ma
                                  University
```

5. Program to implement the 2D Translation, 2D Rotation and 2D scaling.

Two Dimensional Objects

There are three basic transformations that can be applied over objects namely the Translation, Rotation and the Scaling. Other transformations like reflection and shear can also be considered for object transformations. These transformations can be applied over two dimensional objects.

Algorithm

A point (x1, y1) is transformed to location (x2,y2) after applying the basic transformations.

Translation

$x_2 = x_1 + t_x$

$y_2 = y_1 + t_y$

Scaling

$x_2 = a_x*x_1$

$y_2 = a_y*y_1$

Rotation

$x_2 = x_1*\cos(a) + y_1*\sin(a)$

$y_2 = - x_1*\sin(a) + y_1*\cos(a)$

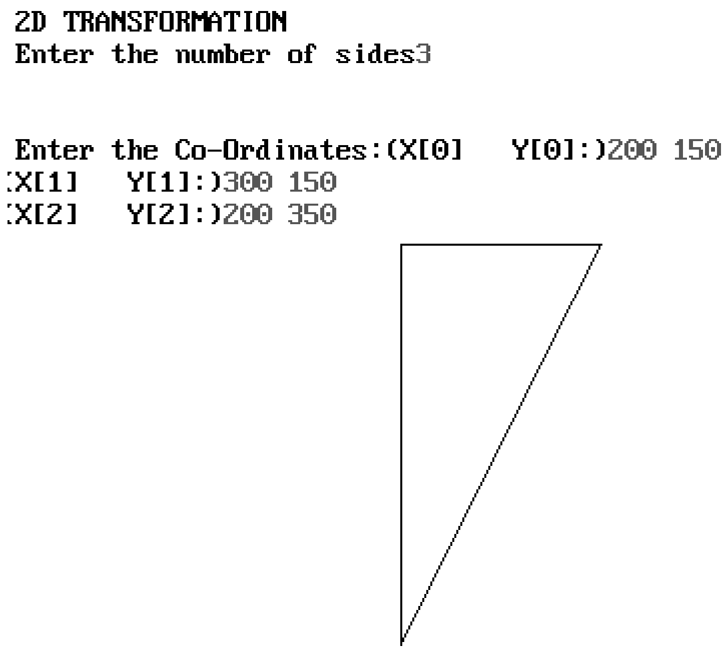
All the above transformations are implemented as matrix multiplication for effective and quick computation. Scaling and rotation are with respect to origin. If scaling is respect to a fixed point (xf, yf) then composite matrix is constructed by first applying translation so that the fixed point coincides with origin then scaling and then perform inverse translation. The three matrices are multiplied to get the final composite matrix. Here the order of multiplication is important. Similarly for rotation about a reference point (xr,yr) the composite matrix is constructed by applying translation so that the reference point coincides with origin then rotation and then inverse translation is performed.

```
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void draw2d(int, float[],float[],float,float);
void main()
{
int gd=DETECT, gm;
float x[10],y[10],x1[10],y1[10],x2[10],y2[10];
float tx,ty,sx,sy,degree;
int option, i,nofsides;
float radius;
initgraph(&gd,&gm,"s:\\cpp\\bgi");
printf("\n\n 2D TRANSFORMATION");
printf("\n Enter the number of sides");
scanf("%d",&nofsides);
printf("\n\n Enter the Co-Ordinates:");
for(i=0;i<nofsides;i++)
{
printf("(X[%d]\tY[%d]:)",i,i);
scanf("%f%f",&x[i],&y[i]);
```

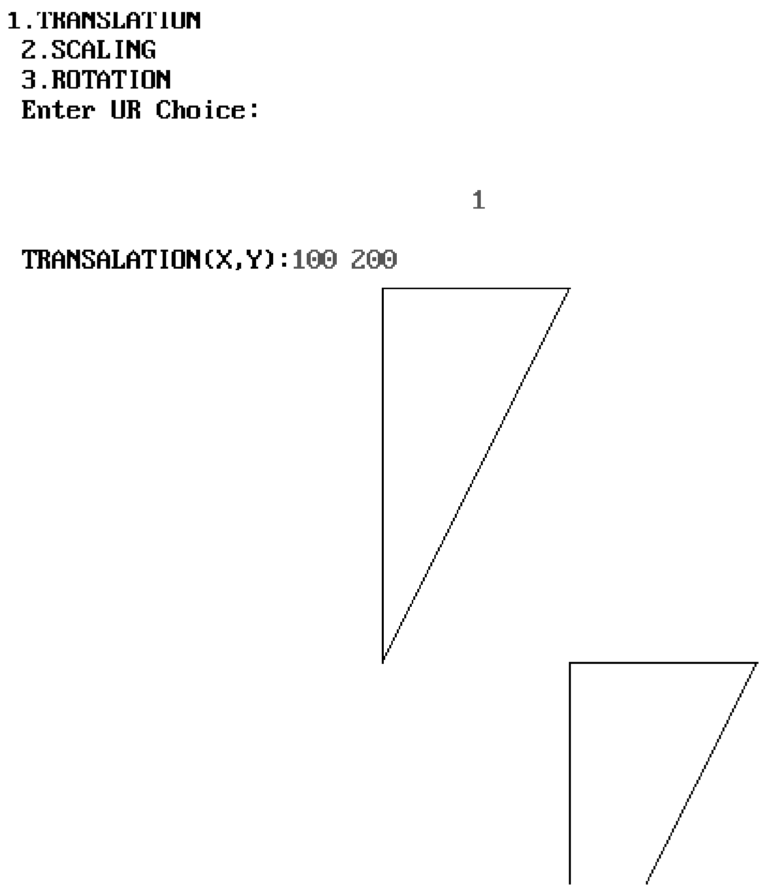
```
}
draw2d(nofsides,x,y,0,0);
getch();
do
{
closegraph();
initgraph(&gd,&gm,”s:\\cpp\\bgi”);
printf(“1.TRANSLATION \n 2.SCALING\n 3.ROTATION”);
printf(“\n Enter UR Choice:”);
scanf(“%d”,&option);
switch(option)
{
case 1:
printf(“\n TRANSALATION(X,Y):”);
scanf(“%f%f”,&tx,&ty);
draw2d(nofsides,x,y,0,0);
draw2d(nofsides,x,y,tx,ty);
break;
case 2:
printf(“\n SCALING FACTORY(X,Y):”);
scanf(“%f%f”,&sx,&sy);
for(i=0;i<nofsides;i++)
{
x1[i]=x[i]*sx;
y1[i]=y[i]*sy;
}
draw2d(nofsides,x,y,0,0);
draw2d(nofsides,x1,y1,0,0);
break;
case 3:
printf("\n ROTATION ANGLE");
scanf("%f",&degree);
radius=(22/7)*degree/180;
x2[0]=x[0];
y2[0]=y[0];
for(i=0;i<nofsides;i++)
{
x2[i]=x[0]+((x[i]-x[0])*cos(radius))-((y[i]-y[0])*sin(radius));
y2[i]=y[0]+((x[i]-x[0])*sin(radius))-((y[i]-y[0])*cos(radius));
}
draw2d(nofsides,x,y,0,0);
draw2d(nofsides,x2,y2,0,0);
break;
default:
exit(0);
}
getch();
}
```

```
while(option<=3);
}
void draw2d(int nofs,float x[20],float y[20],float tx, float ty)
{
int i;
for(i=0;i<nofs;i++)
{
if(i!=(nofs-1))
line (x[i]+tx,y[i]+ty,x[i+1]+tx,y[i+1]+ty);
else
line(x[i]+tx,y[i]+ty,x[0]+tx,y[0]+ty);
}
}
```

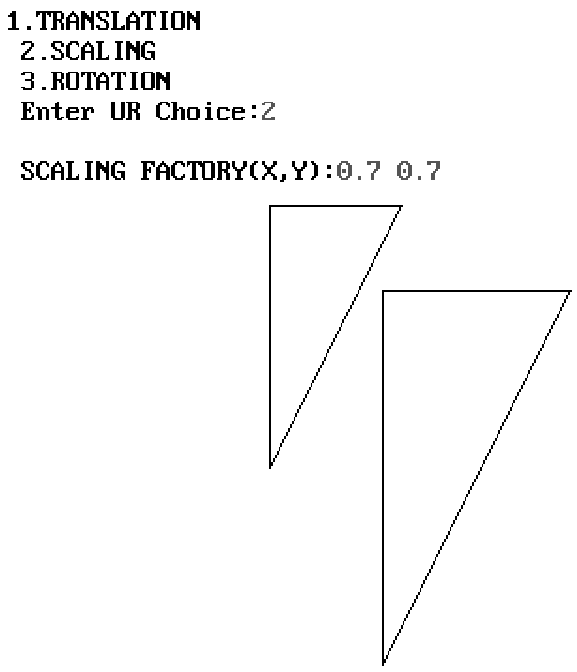
OUTPUT:



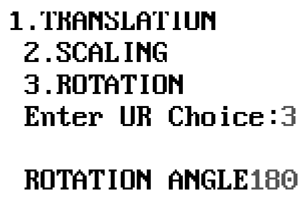
```
2D TRANSFORMATION
Enter the number of sides3


Enter the Co-Ordinates:(X[0]   Y[0]:)200 150
:X[1]   Y[1]:)300 150
:X[2]   Y[2]:)200 350
```



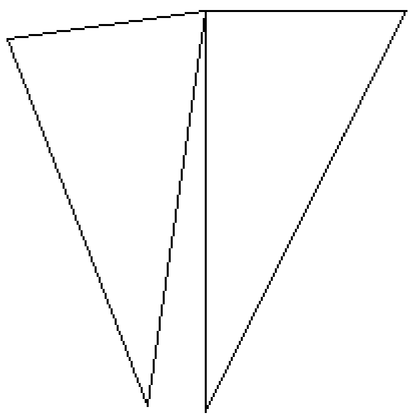
```
1.TRANSLATION
 2.SCALING
 3.ROTATION
 Enter UR Choice:

                              1

 TRANSALATION(X,Y):100 200
```



```
1.TRANSLATION
 2.SCALING
 3.ROTATION
 Enter UR Choice:2

 SCALING FACTORY(X,Y):0.7 0.7
```



```
1.TRANSLATION
 2.SCALING
 3.ROTATION
 Enter UR Choice:3

 ROTATION ANGLE180
```



6. Program to implement the 3D Translation, 3D Rotation and 3D scaling.

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx,maxy,midx,midy;

void axis()

{

getch();

cleardevice();

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}
```

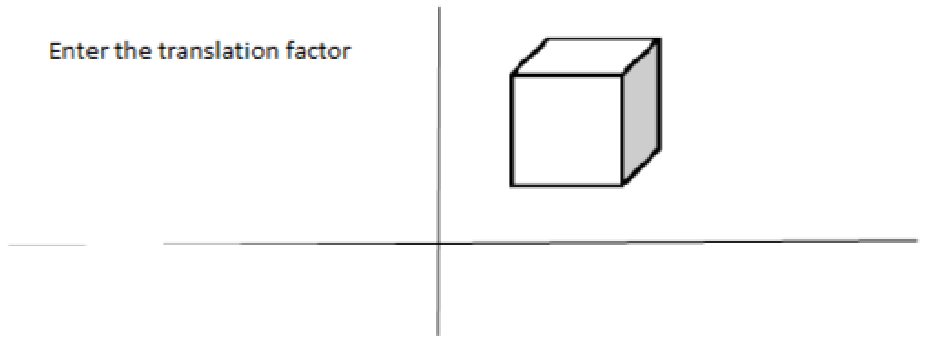
```
void main()
{
int gd,gm,x,y,z,o,x1,x2,y1,y2;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"s:\\cpp\\bgi");
setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
axis();
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
printf("\Enter the Translation Factor");
scanf("%d%d",&x,&y);
axis();
printf("After translation");
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
bar3d(midx+x+100,midy-(y+150),midx+x+60,midy-(y+100),10,1);
axis();
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
printf("Enter the Scalling Factor");
scanf("%d%d%d",&x,&y,&z);
axis();
printf("After Scalling");
```

}

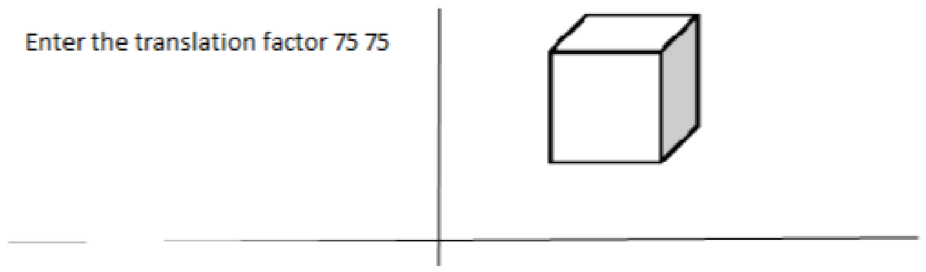
```
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
bar3d(midx+(x*100),midy-(y*150),midx+(x*60),midy-(y*100),10*z,1);
axis();
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
printf("Enter the Rotation Angle");
scanf("%d",&o);
x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);
y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);
axis();
printf("After Rotating About Z-axis");
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
bar3d(midx+x1,midy-y1,midx+x2,midy-y2,10,1);
axis();
printf("After Rotating X-axis");
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
bar3d(midx+100,midy-x1,midx+60,midy-x2,10,1);
axis();
printf("After Rotating Y-axis");
bar3d(midx+100,midy-150,midx+60,midy-100,10,1);
bar3d(midx+x1,midy-150,midx=x2,midy-100,10,1);
getch();
closegraph();
}
```

OUTPUT:

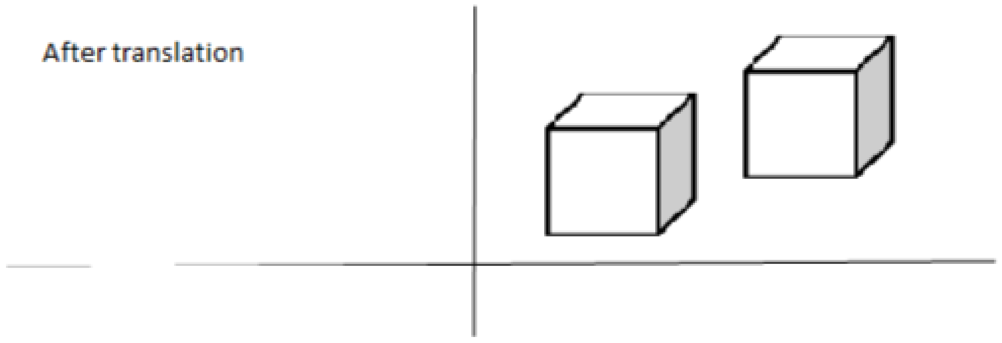
3D Object Transformations



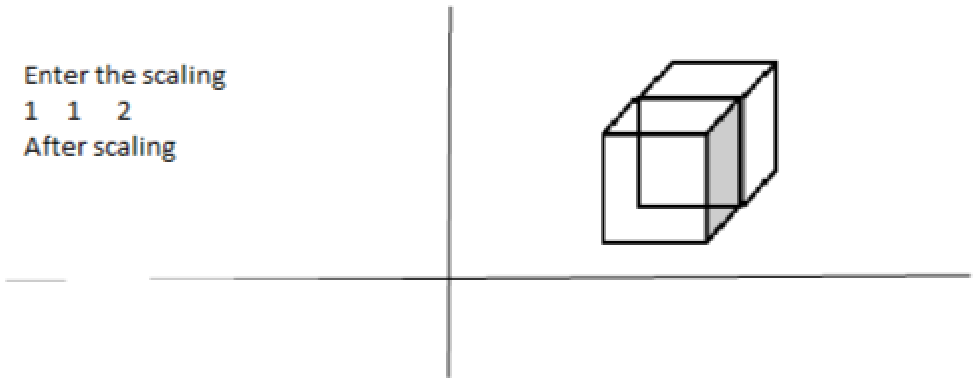
```
Enter the translation factor
```



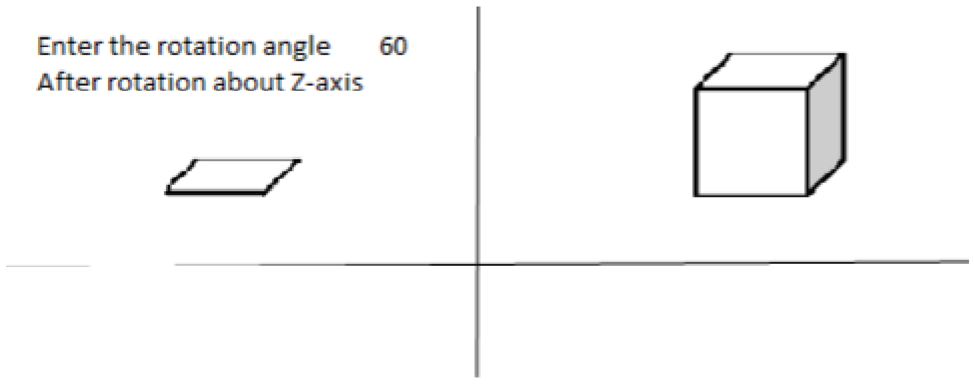
```
Enter the translation factor 75 75
```



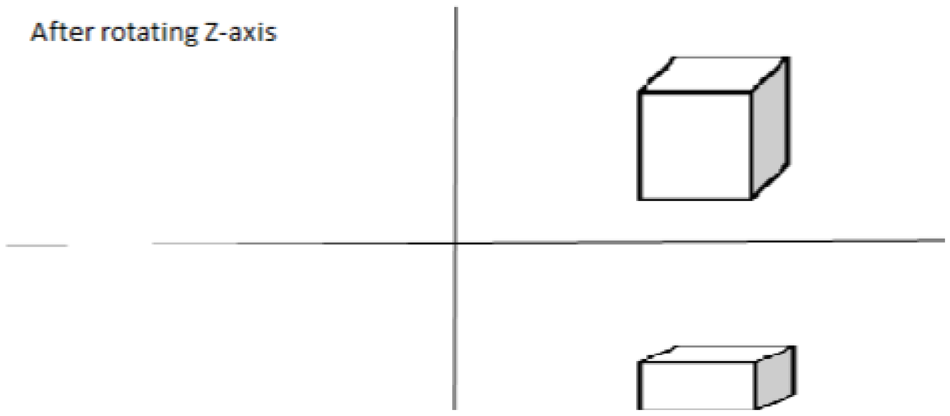
```
After translation
```



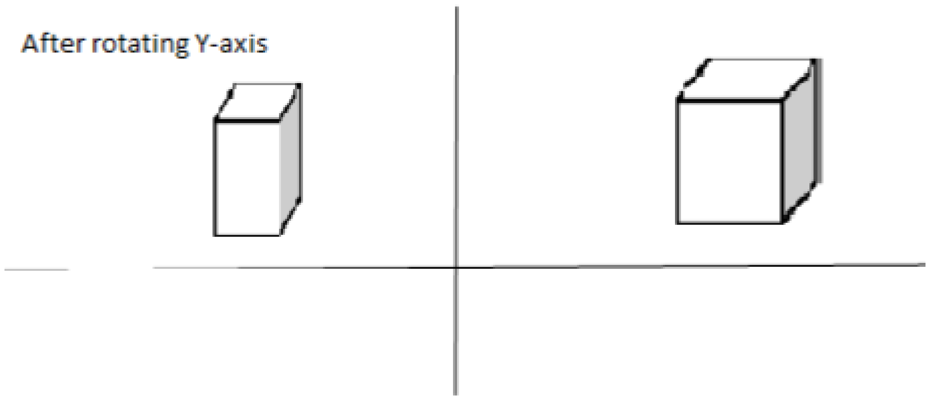
```
Enter the scaling
1   1    2
After scaling
```



```
Enter the rotation angle      60
After rotation about Z-axis
```



```
After rotating Z-axis
```



```
After rotating Y-axis
```

7. Program to implement the Shearing and Reflection of an object.

Shearing

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called shear.

Reflection

A reflection is a transformation that produces a mirror image of an object. The mirror image for a two-dimensional reflection is generated relative to an axis of reflection by rotating the object 180 degree about the reflection axis.

A) SHEARING :

```
**********
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<stdlib.h>
voidmul(int mat[3][3],int vertex[10][3],int n);
void shear(int vertex[10][3],int n);
voidinit(int vertex[10][3],int n);
int main()
{
inti,x,y;
int vertex[10][3],n;
clrscr();
printf("\nEnter the no. of vertex : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the points (x,y): ");
scanf("%d%d",&x,&y);

vertex[i][0]=x;

vertex[i][1]=y;

vertex[i][2]=1;

}

shear(vertex,n);

getch();

return 0;

}

voidinit(int vertex[10][3],int n)

{

intgd=DETECT,gm,i;

initgraph(&gd,&gm,"s:\\cpp\\bgi");

setcolor(10);

line(0,240,640,240);      //drawing X axis

line(320,0,320,480);      //drawing Y axis

setcolor(3);

line(450,20,490,20);

setcolor(15);

line(450,50,490,50);

setcolor(6);

outtextxy(500,20,"Original");

outtextxy(500,50,"Transformed");

setcolor(3);
```

```
for(i=0;i<n-1;i++)

{

line(320+vertex[i][0],240-vertex[i][1],320+vertex[i+1][0],240-vertex[i+1][1]);

}

line(320+vertex[n-1][0],240-vertex[n-1][1],320+vertex[0][0],240-vertex[0][1]);

}

voidmul(int mat[3][3],int vertex[10][3],int n)

{

inti,j,k;

int res[10][3];

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

{

for(j=0;j<3;j++)

{

res[i][j]=0;

for(k=0;k<3;k++)

{

res[i][j] = res[i][j] + vertex[i][k]*mat[k][j];

}

}

}

setcolor(15);

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

{
```

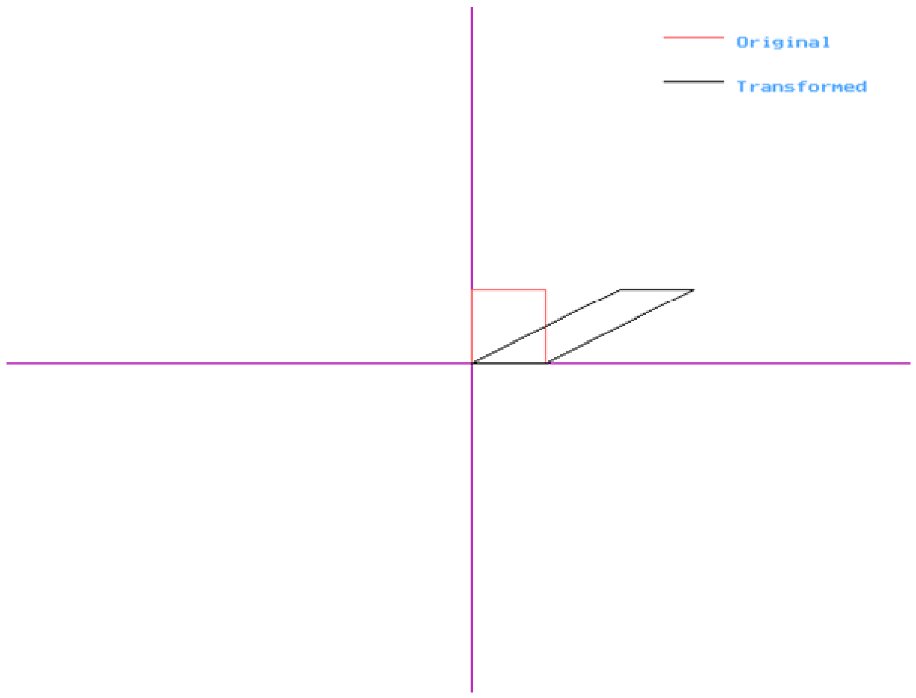
```
line(320+res[i][0],240-res[i][1],320+res[i+1][0],240-res[i+1][1]);

}

line(320+res[n-1][0],240-res[n-1][1],320+res[0][0],240-res[0][1]);

}

void shear(int vertex[10][3],int n)

{

intopt,xsh,ysh;

intshear_array[3][3];

printf("\n1.x-shear\n2.y-shear\nYour Choice: ");

scanf("%d",&opt);

//initializing the shearing transformation matrix as per the required direction

switch(opt)

{

case 1:

printf("\nEnter the x shear : ");

scanf("%d",&xsh);

//values for X shear

shear_array[0][0]=1;

shear_array[1][0]=xsh;

shear_array[2][0]=0;

shear_array[0][1]=0;

shear_array[1][1]=1;

shear_array[2][1]=0;

shear_array[0][2]=0;
```

```
    shear_array[1][2]=0;
    shear_array[2][2]=1;
    //initializing the graphics mode and drawing the original object
    init(vertex,n);
    //multiplying the object with shearing transformation matrix and displaying the sheared
image
    mul(shear_array,vertex,n);
    break;
    case 2:
    printf("\nEnter the y shear : ");
    scanf("%d",&ysh);
    //values for Y shear
    shear_array[0][0]=1;
    shear_array[1][0]=0;
    shear_array[2][0]=0;
    shear_array[0][1]=ysh;
    shear_array[1][1]=1;
    shear_array[2][1]=0;
    shear_array[0][2]=0;
    shear_array[1][2]=0;
    shear_array[2][2]=1;
    //initializing the graphics mode and drawing the original object
    init(vertex,n);
    //multiplying the object with shearing transformation matrix and displaying the sheared
image
```

```
mul(shear_array,vertex,n);

break;

}

}
```

Output:

```
Enter the no. of vertex : 4

Enter the points (x,y): 0 0

Enter the points (x,y): 50 0

Enter the points (x,y): 50 50

Enter the points (x,y): 0 50

1.x-shear
2.y-shear
Your Choice:

1

Enter the X shear: 2
```



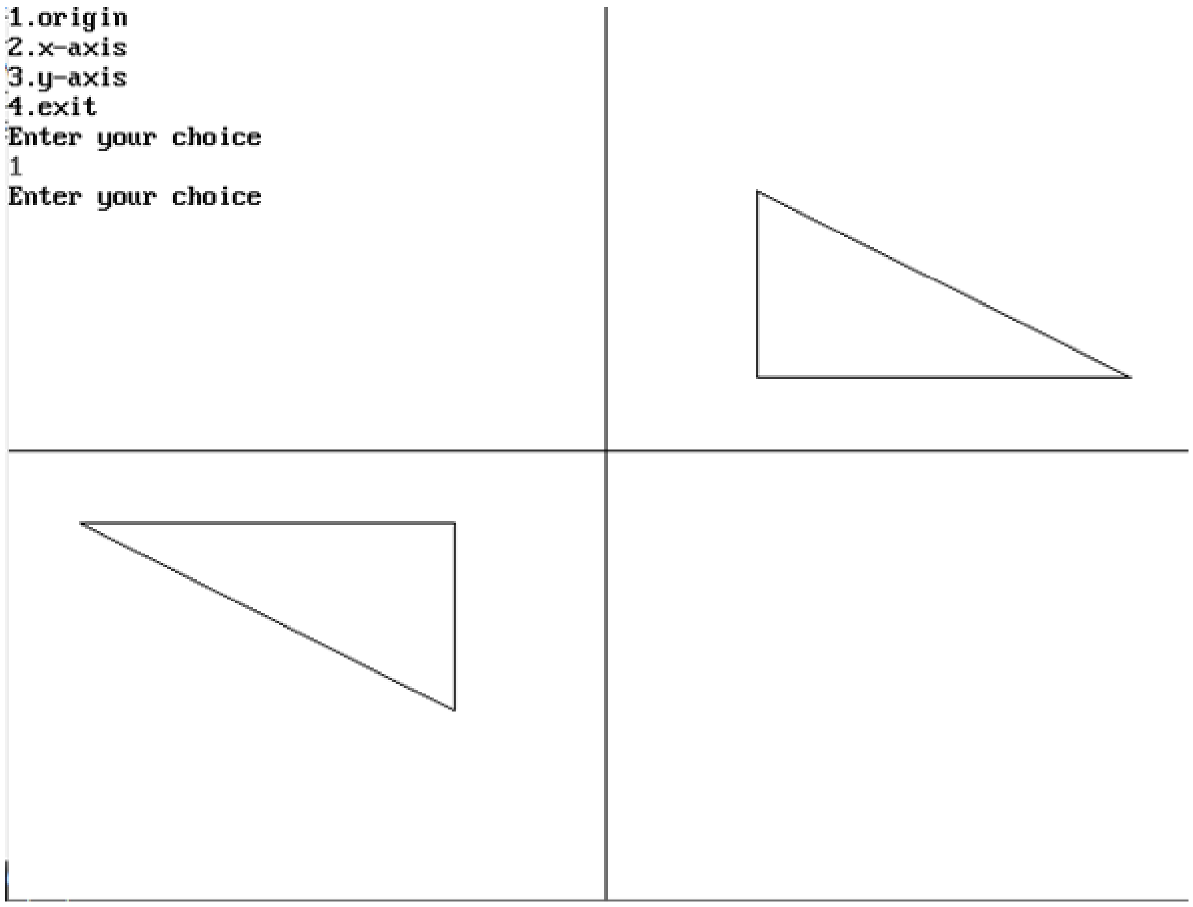
B) REFLECTION:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int a,a1,b,b1,c,c1,xt,ch;

intgd=DETECT,gm;

initgraph(&gd,&gm,”s:\\cpp\\bgi”);

a=getmaxx();

a1=getmaxy();

b=a/2;

b1=a1/2;

line(b,0,b,a1);

line(0,b1,a,b1);

line(400,200,600,200);

line(400,200,400,100);

line(400,100,600,200);

printf(“1.origin\n”);

printf(“2.x-axis\n”);

printf(“3.y-axis\n”);

printf(“4.exit\n”);
```

```
do

{

printf("Enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

c=400-b;c1=200-b1;

line(b-c,b1-c1,b-c-200,b1-c1);

line(b-c,b1-c1,b-c,b1-c1+100);

line(b-c,b1-c1+100,b-c-200,b1-c1);

break;

case 2:

c=400-b;c1=200-b1;

line(b+c,b1-c1,b+c+200,b1-c1);

line(b+c,b1-c1,b+c,b1-c1+100);

line(b+c,b1-c1+100,b+c+200,b1-c1);

break;

case 3:

c=400-b;c1=200-b1;

line(b-c,b1+c1,b-c-200,b1+c1);

line(b-c,b1+c1,b-c,b1+c1-100);
line(b-c,b1+c1-100,b-c-200,b1+c1);

break;

}

}while(ch<4);

getch();

closegraph();

}
```

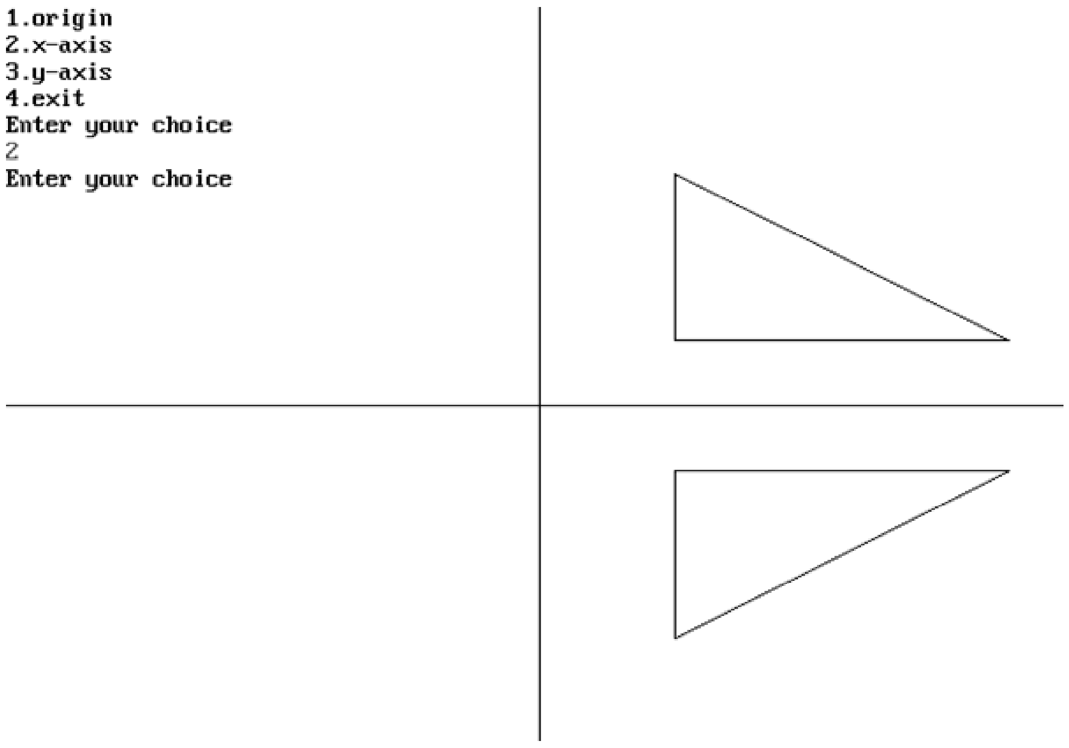
OUTPUT:

Enter your choice:1



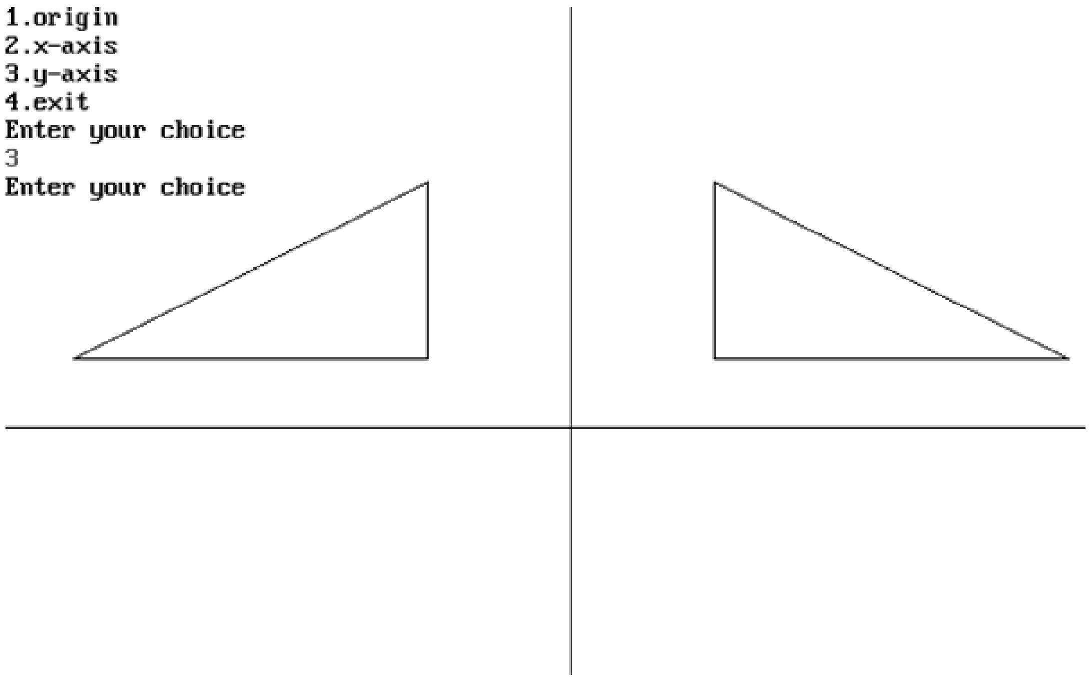
```
1.origin
2.x-axis
3.y-axis
4.exit
Enter your choice
1
Enter your choice
```

Enter your choice:2



```
1.origin
2.x-axis
3.y-axis
4.exit
Enter your choice
2
Enter your choice
```

Enter your choice:3



```
1.origin
2.x-axis
3.y-axis
4.exit
Enter your choice
3
Enter your choice
```