D. Y. PATIL COLLEGE OF ENGINEERING

Akurdi, Pune - 411 044.

LABORATORY JOURNAL

DYP
DR. D. Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

SUBJECT : OOP & Computer Graphics Laboratory

INDEX

Sr. No.	Date of Performance	Title of the Experiment	Page No.	Remarks
1.	30/8/22	Draw pattern using DDA & Bresenham Algorithm	1	
2.	6/9/22	Create a student database information system using concept of OOP	9	
3.	13/9/22	Draw a concave polygon & fill it using scan-fill algorithm	17	
4.	20/9/22	Create a C++ program for a publishing company & apply concept of inheritance	24	
5.	27/9/22	Draw pattern line-clipping Cohen-Sutherland algorithm	34	
6.	18/10/22	Draw a 2D object & perform following basic transformations scaling, translation, rotation	39	

Certificate

Certified that Mr./~~Miss~~ Atharv Gupta

Roll No. SEC02223A068 of SE-'A' class has carried out above practical/~~term work~~

within the four walls of the college. His/~~Her~~ performance is satisfactory & attendance is 100 %

21/12/22

Date | Staff Member In - Charge | Head of the Department | Principal

HEAD
Department of Computer Engineering
D. Y. Patil College of Engineering,
Akurdi, Pune-411 044

Supplied by : Dr. D. Y. Patil Pratishthan Educational Complex, Student Consumer Store, Akurdi, Pune - 44.

D. Y. PATIL COLLEGE OF ENGINEERING

Akurdi, Pune - 411 044.

LABORATORY JOURNAL

DYP
DR. D. Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

SUBJECT : OOPCGL

INDEX

Sr. No.	Date of Performance	Title of the Experiment	Page No.	Remarks
7.	19/10/22	Implement a program for complex numbers using operator overloading	46	
8.	9/11/22	Implement the concept of file handling	51	
9.	15/11/22	Write a program to implement the game of TicTacToe	55	
10.	22/11/22	Write a function template for selection sort.	59	
11.	29/11/22	Implement the concept of STL for sorting & searching user defined records using vector.	64	
12.	5/12/22	Write a program in C++ to use map associative container	69	

Certificate

Certified that Mr./~~Miss~~ Atharv Gupta

Roll No. SEC02223A068 of SE-'A' class has carried out above practical/~~term work~~

within the four walls of the college. His/~~Her~~ performance is satisfactory & attendance is 100 %

/12/22

Date

Staff Member In - Charge | Head of the Department | Principal

Supplied by : Dr. D. Y. Patil Pratishthan Educational Complex, Student Consumer C

D. Y. PATIL COLLEGE OF ENGINEERING

Akurdi, Pune - 411 044.

LABORATORY JOURNAL

DYP
DR. D. Y. PATIL EDUCATIONAL COMPLEX
AKURDI, PUNE

SUBJECT : DDPCGL

INDEX

Sr. No.	Date of Performance	Title of the Experiment	Page No.	Remarks
13.	7/12/22	generate fractals using Koch curves	73	
14.	9/12/22	Write an OpenGL program to demonstrate sunrise & sunset.	77	
15.	9/12/22	MINI PROJECT To implement a working Analog clock using graphics library with concepts of OOP.	82 ⋮ 90	

Certificate

Certified that Mr./~~Miss~~ Atharv gupta

Roll No. SEC02223A068 of SE-'A' class has carried out above practical/~~term work~~ within the four walls of the college. His/Her performance is satisfactory & attendance is 100 %

/12/22
Date

Staff Member In - Charge | Head of the Department | Principal

Supplied by : Dr. D. Y. Patil Pratishthan Educational Complex, Student Consumer Store, Akurdi, Pune - 44.

Assignment-A3

→ **Title**

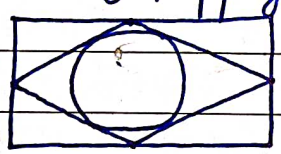
To draw a pattern using lines and circles by Bresenham's line drawing algorithm and digital differential analyzer.

→ **Objectives**

To understand basics of computer graphics, apply various methods & techniques for implementing line & circle drawing algorithms using concepts of Object Oriented Programming

→ **Problem statement**

Write a C++ program to draw the following pattern. Use DDA line & Bresenham's circle drawing algorithms. Apply the concept of encapsulation.



→ **Outcomes**

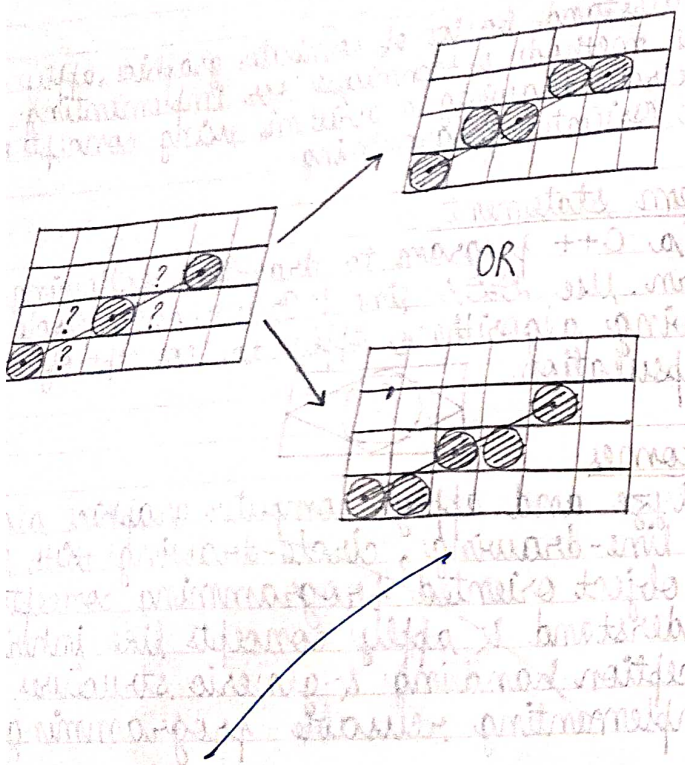
i) Analyze and apply computer graphics algorithms for line-drawing, circle-drawing with the help of object oriented programming concepts.

ii) Understand & apply concepts like inheritance, exception handling & generic structures for implementing reusable programming codes.

→ **Software & Hardware Requirements**

i) **Software**: Ubuntu/Linux OS, Open source C++, Programming tool like Lf++, GCC.

ii) **Hardware**: Processor → intel(i5)
storage → 1 TB
RAM → 4 GB

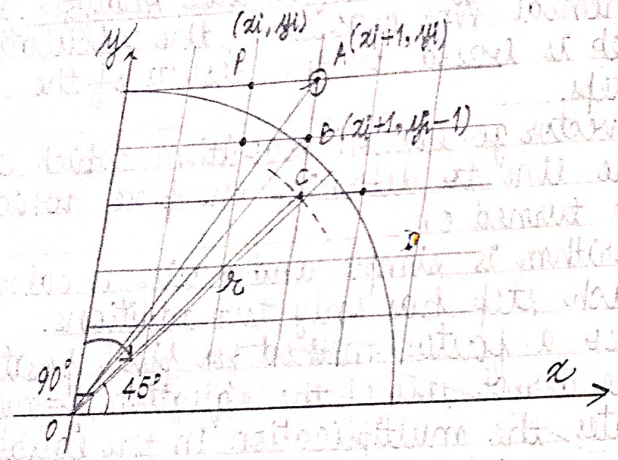


→ Theory

I. DDA Line Drawing Algorithm

In computer graphics, we need to represent continuous graphics objects using discrete pixels. This process is known as scan conversion. Every graphics system must transform the primitives like lines, circles & ellipses into a collection of pixels. Line drawing algorithms are used to draw a line in discrete graphical media.

- DDA stands for Digital Differential Analyzer. It is an incremental line algorithm, the calculation of each step is based on the results of the previous steps.
- It is a vector generation algorithm which steps along the line to determine the pixel which should be turned on.
- DDA Algorithm is simple and easier to calculate since each step has only two additions.
- It is also a faster method for pixel positions than the direct use of the equation $y = mx + b$. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to find the pixel positions along the line path.
- One of the disadvantages of this algorithm is that floating point arithmetic is still time-consuming. Round-off operation consumes a lot of time & accumulations of rounding off values causes accumulation error. End point accuracy is also poor.

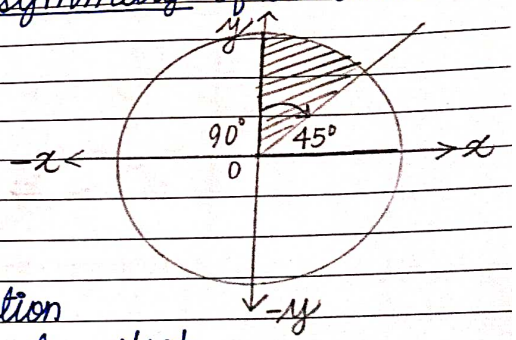


Scan Conversion with Bresenham's Algorithm

II. Bresenham's Circle Drawing Algorithm

If a circle is to be plotted efficiently, the use of trigonometric & power functions must be avoided. It is desirable to perform the calculations necessary to find the scan converted points with only integer addition, subtraction, and multiplication by powers of 2. Bresenham's circle drawing algorithm allows these goals to be met.

- The Bresenham's circle drawing algorithm considers the eight-way symmetry of the circle to generate it. It plots 1/8th part of the circle, i.e. from 90° to 45°, the x moves in positive direction and the y moves in negative direction.

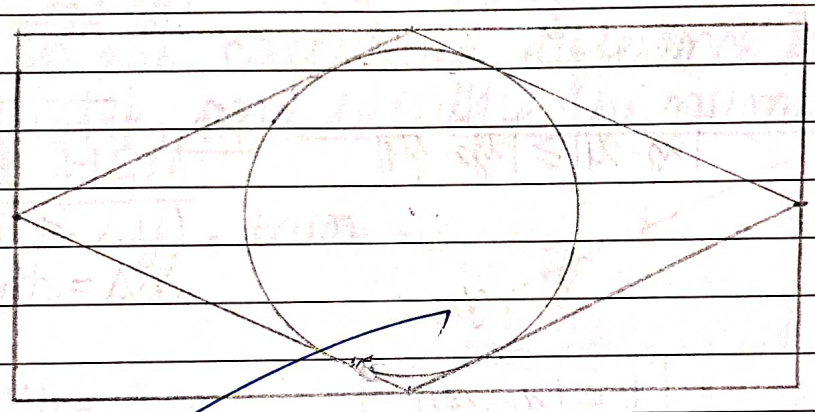


- To achieve best approximation to the true circle, we have to select the pixels in the raster which fall the least distance from the true circle.
- If points are generated from 90° to 45° portion of the circle, each new point closest to the true circle can be found by applying either of the two options:

(i) Increment in positive x-direction by one unit or
(ii) Increment in positive x-direction and negative y-direction by, both one unit.

So, a method of selecting between these two choices is all that is necessary to find the points.

To draw this pattern, we need two rectangles and one circle. We can use suitable geometry to get coordinates to draw rectangles and the circle. Then, we can call DDA line drawing algorithm & Bresenham's circle drawing algorithm by passing appropriate parameters to get the required pattern. We can also ask for the dimensions from the user, but not here, since it is a predefined pattern.



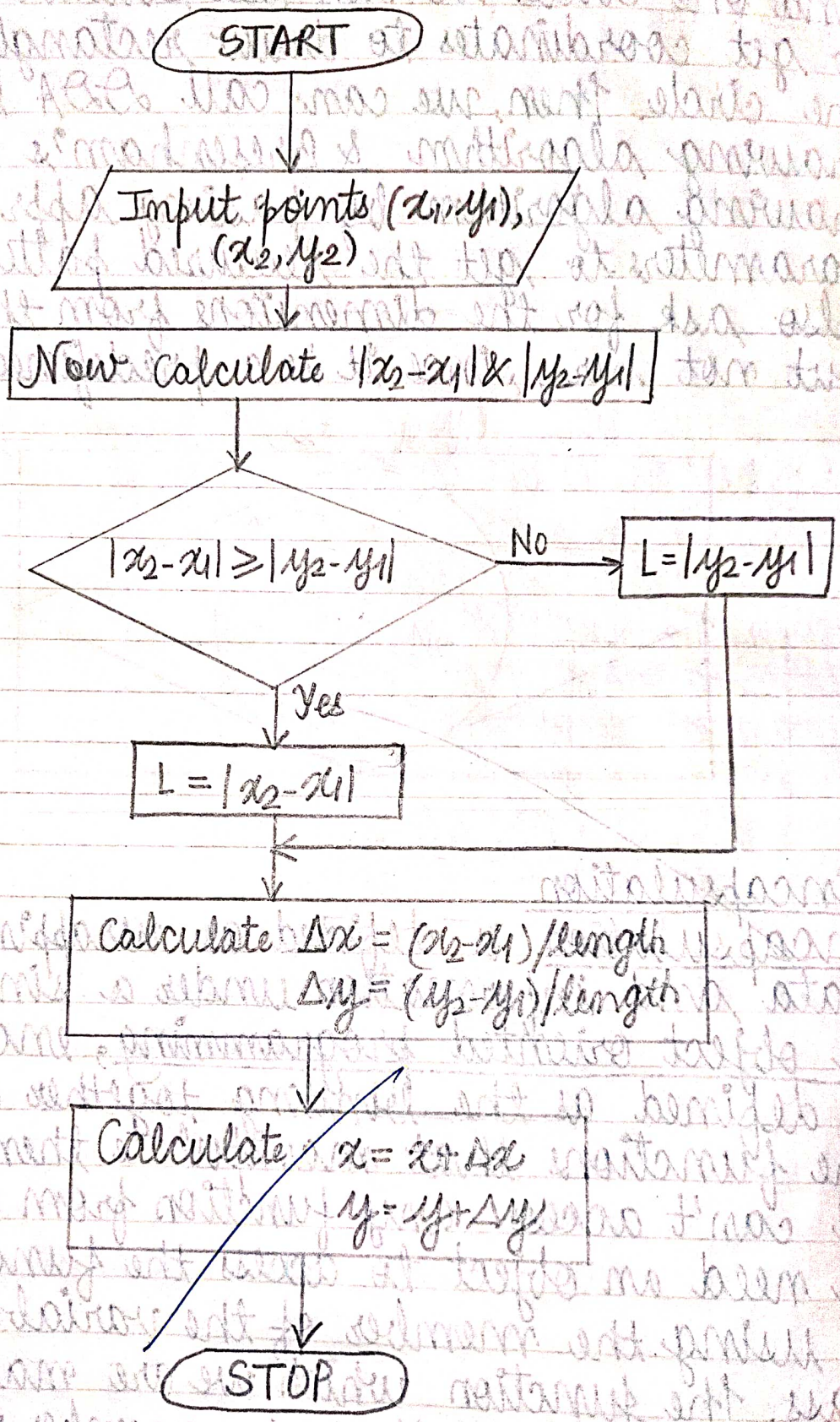
→ Encapsulation

Encapsulation is defined as wrapping up of data and information under a single unit.

- In object oriented programming, encapsulation is defined as the binding together of data & the functions that manipulate them.
- We can't access any function from class directly. We need an object to access the function which is using the member of the variable of that class. The function which we are making inside the class must use all of the member variables, then only it is called encapsulation

→ Flowchart

DDA Line Drawing Algorithm



→ Algorithms

I. DDA Line-Drawing Algorithm

i) START

ii) Read the line end points (x_1, y_1) and (x_2, y_2) such that they are not equal. If found equal, then plot that point & exit.

iii) Calculate the horizontal & vertical difference between these two end points.

$\Delta x = |x_2 - x_1|$

$\Delta y = |y_2 - y_1|$

iv) Based on the calculated difference in the previous step, and identify the number of steps to put pixel.

if ($\Delta x \geq \Delta y$), then

length = Δx

else

length = Δy

end if

v) Calculate the increment in x-coordinate & y-coordinate.

$\Delta x = (x_2 - x_1)/\text{length}$

$\Delta y = (y_2 - y_1)/\text{length}$

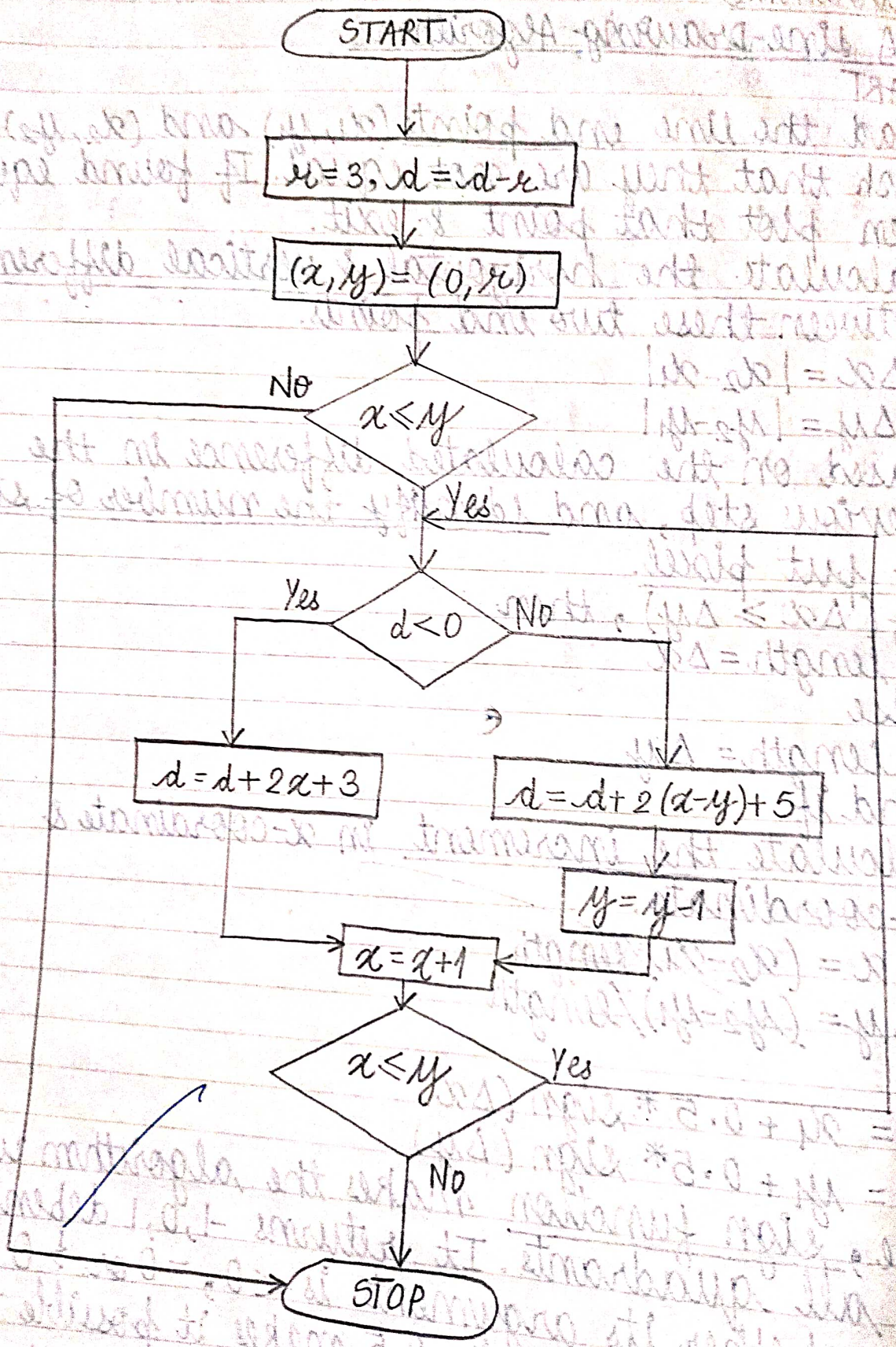
vi) $x = x_1 + 0.5 * \text{sign}(\Delta x)$

$y = y_1 + 0.5 * \text{sign}(\Delta y)$

Here, sign function makes the algorithm work in all quadrants. It returns -1, 0, 1 depending on whether its argument is < 0, $= 0$ or > 0 respectively. The factor 0.5 makes it possible to round the values in the integer function rather than truncating them

→ Flowchart

Bresenham's Circle Drawing Algorithm



plot(Integer(x), Integer(y))

vii) Put the pixel by successfully implementing x & y coordinates accordingly and complete drawing of the line.

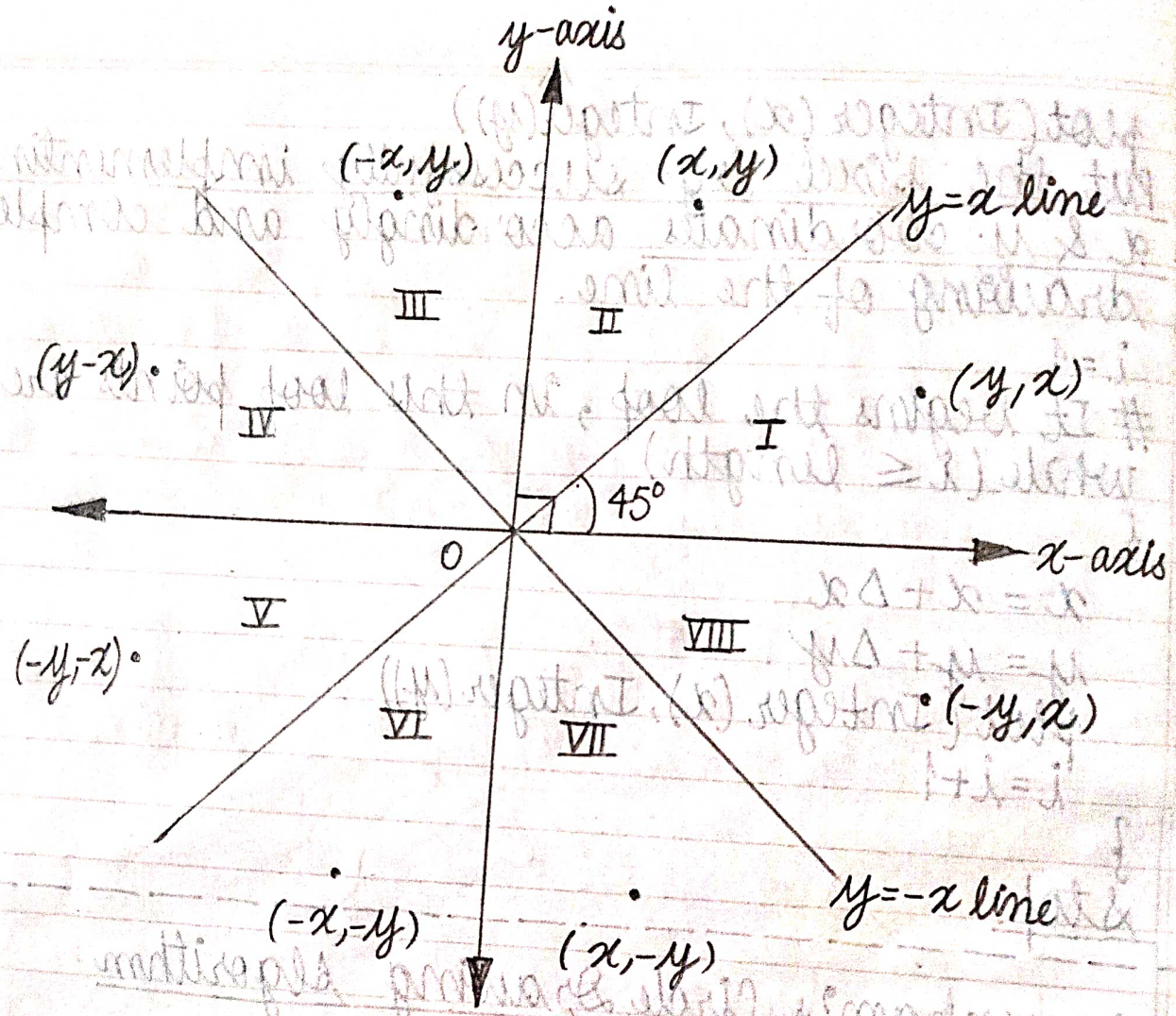
```
i=1
# It begins the loop, in this loop points are plotted
while (i ≤ length)
{
  x = x + Δx
  y = y + Δy
  plot(Integer(x), Integer(y))
  i = i+1
}
```

viii) Stop.

II. Bresenham's Circle Drawing algorithm

i) Start
ii) Read the radius (r) of the circle
iii) d = 3 - 2r # initialize the decision variable
iv) x = 0, y = r # initialize starting point
v)

```
do
{
  plot (x, y)
  if (d<0) then
  {
    d = d + 4x + 6
  }
else
  {
    d = d + 4(x-y) + 10
```


y-axis
(-x,y)
(x,y)
y=x line
III
II
(y-x)
(y,x)
IV
I
45°
O
x-axis
V
VIII
(-y,-x)
(-y,x)
VI
VII
(-x,-y)
(x,-y)
y=-x line

$y = y - 1$
}
$x = x + 1$
} while($x < y$)

vi) stop

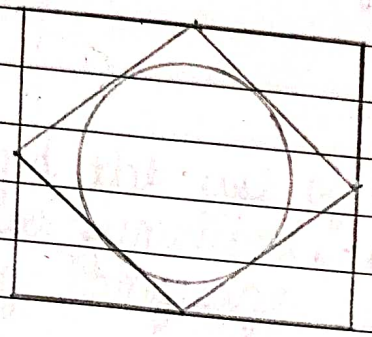
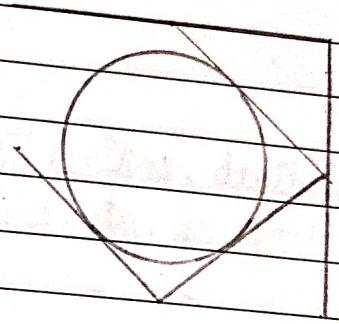
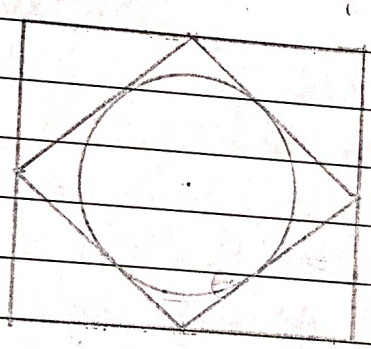
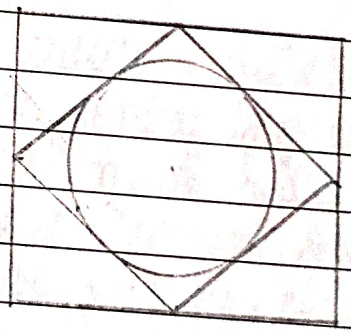
→ Here in step 5, the computer will give the output for octant-1 of the circle with radius 'r' after the completion of all the iterations of the do-while loop

- Because of the <u>eight-way symmetry</u> property of the circle, we can draw other octants of the circle using putpixel() functions:-

Octant 1 - putpixel (centx + x, centy - y, ColorName);
Octant 2 - putpixel (centx + y, centy - x, ColorName);
Octant 3 - putpixel (centx + y, centy + x, ColorName);
Octant 4 - putpixel (centx + x, centy + y, ColorName);
Octant 5 - putpixel (centx - x, centy + y, ColorName);
Octant 6 - putpixel (centx - y, centy + x, ColorName);
Octant 7 - putpixel (centx - y, centy - x, ColorName);
Octant 8 - putpixel (centx - x, centy - y, ColorName);

21/10/22

→ Test Cases

S. no.	Step	Expected output	Actual output	Result
1.	Print the pattern			Fail
2.	Print the pattern			Pass

→ Conclusion

Hence, we have successfully understood DDA line drawing & Bresenham's circle drawing algorithms & how to implement them to draw various patterns on the output screen.

OOP
Assignment - A2

→ **Title**

To display personal information using constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, etc.

→ **Objectives**

To understand the use of constructors, destructors, static member functions, pointers, etc. in C++ programming language.

→ **Problem statement**

Develop a program in C++ to create a database of student's information system containing the following information: Name, roll number, Class, Division, Date of Birth, Blood group, contact Address, Telephone number, Driving license number, etc. Construct the database with suitable member functions. Make use of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators - new & delete as well as exception handling

→ **Outcomes**

Understand and apply the concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

→ Software & Hardware Requirements

i) Software : Ubuntu/Linux OS, Open source C++, Programming tool like Gt++, GCC.

ii) Hardware : Processor → intel (i5)
Storage

→ Theory

- Constructors
Objects need to initialize the variables. such, when not initialized, they hold some garbage value. And if such variables are taken for operation, then unexpected results may occur as initialized variables are used for processing. In order to avoid that, the class can include a special function called constructor.
- The constructor can be automatically called whenever a new object of this class is created.
- The constructor will have the same name as the class name.
- It should not have any return type.
- The constructor must be declared in public mode.
- The constructors are not inherited. But the derived class can invoke the constructor of the base class.
- Syntax

```
class example
    {
     public:
        example() {
                    // declaration & definition of constructor
                    }
    };
```

Various types of constructors used in C++ are-

i) Default Constructor
ii) Parameterized constructor
iii) Default argument constructor
iv) copy constructor

→ Destructor

A destructor is called when the object is destroyed. The object can be destroyed automatically when the scope of the objects ends (i.e. when the function ends) or explicitly by using operator delete.

- The destructor must also have the same name as the class, but preceded with a tilde sign (~) and it must also return no value.
- We normally use the destructor when the object assigns dynamic memory in its lifetime.
- syntax

```
~class_name()
  {
   -----
  }
```

→ static member functions

static member functions can access only static members. The members of the class must be declared with the keyword 'static'.

- When the keyword static is associated with any class variable then that means only a single copy of that variable must be created & all other objects must share that copy.

- The static data members can be accessed without any instantiation of the class as object. All static variables are initialized to zero before first object is created.
- Syntax

```
class test
  {
   private:
        static int count; // declaration
            :
  };
int test :: count = 100; // definition without instantiation
                            of class as object.
```

→ Inline function

The inline function is a function whose code is copied in place of each function call. The inline specifies that the compiler should insert the complete body of function in every context where the function is used.

- Syntax

eg.

```
inline largest(int x, int y, int z)
  {
   -----
  }
```

→ Friend function

The friend function is a function that is not a member function of the class but it can access the private & protected members of the class.

- The friend function is given by keyword 'friend'.
- These are special functions which are not declared anywhere in the class but have been given special permission to access the private members of the class.
- syntax

friend return-type function_name(argument(s)).

→ 'this' pointer

A keyword 'this' identifies a special type of pointer. For example, suppose you create an object named 'x' of class 'A' & class A has nonstatic member function f(). If you call the function x.f(), the keyword 'this' in the body of f() stores the address of x. You cannot declare this pointer or make assignments to it.

- A static member function does not have a 'this' pointer.
- The 'this' pointer is passed as a hidden parameter to the member function call and it is available in the function definition as a local variable.

→ 'new' & 'delete' operators for Dynamic Memory

C++ has two operators new and delete that perform the task of allocating & freeing the memory in a better and easier way.

- 'new' operator

The 'new' operator denotes a request for memory allocation on the free store. If sufficient memory is available, a new operator initializes the memory and returns the address of the newly allocated & initialized memory to the pointer variable

- syntax

pointer_variable = new data_type;

- 'delete' operator

Since it is programmer's responsibility to deallocate memory, programmers are provided 'delete' operator in C++ language.

- Syntax

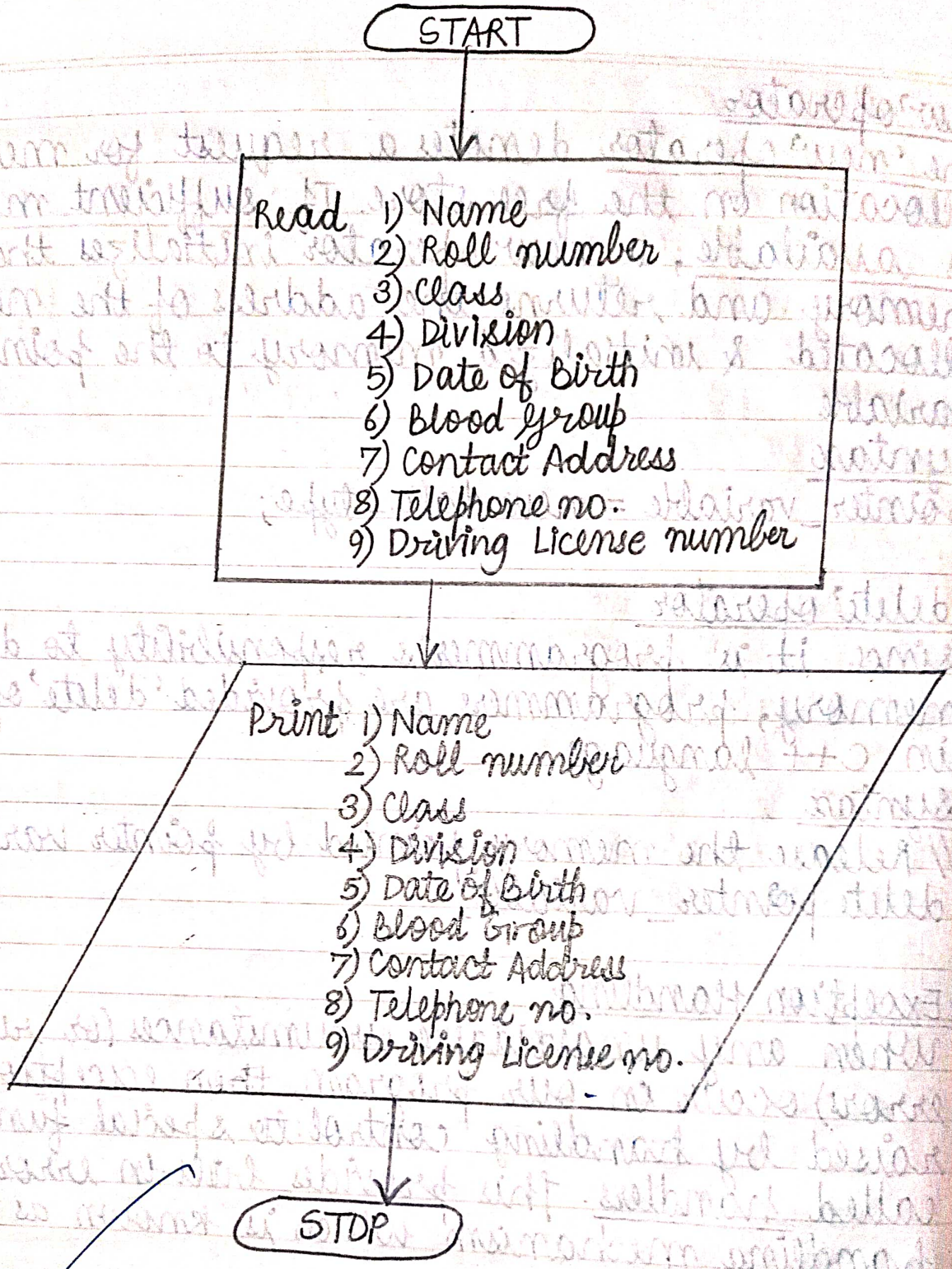
```
// Release the memory pointed by pointer variable
delete pointer_variable;
```

→ Exception Handling

When any unavoidable circumstances (or runtime errors) occur in our program, then exceptions are raised by handling control to special functions called handlers. This provides built-in error handling mechanism which is known as exception handling

- The purpose of exception handling mechanism is to detect and report the exceptional situations so that appropriate action can be taken.

→ Flowchart


START
Read 1) Name
2) Roll number
3) Class
4) Division
5) Date of Birth
6) Blood group
7) Contact Address
8) Telephone no.
9) Driving License number
Print 1) Name
2) Roll number
3) Class
4) Division
5) Date of Birth
6) Blood Group
7) Contact Address
8) Telephone no.
9) Driving License no.
STOP

→ Algorithm

i) Start
ii) Read personal information such as Name, Date of Birth, Blood group, Roll number, class, division, contact address, telephone number, driving license number and others.
iii) Print all information from database
iv) Stop

- Input

Personal information such as Name, Roll number, Class, Division, Date of Birth, Blood group, contact address, telephone number, driving license number.

- Output

Display information from database. The result is in the following format-

Name...
DOB...
⋮
Driving license number...

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Accept the student's information	Enter name: Enter Roll no.: Enter Class: Enter Division: Enter Date of Birth: Enter Blood Group: Enter Address: Enter Phone no.: Enter DL no.:	Same as the expected output	Pass
2.	Display the information	Name: Roll no.: Class: Division: DOB: Blood Group: Address: Phone no.: DL no:	Same as the expected output.	Pass
3.	End program	End	End	Pass

21/10/24

→ Conclusion

Hence, we have successfully studied the concept of constructor, friend class, this pointer, inline code & dynamic memory allocation operators - new & delete.

CG
Assignment - A1

→ **Title**

To draw a concave polygon and fill it with colors using scan-fill algorithm.

→ **Objectives**

To understand basics of computer graphics & polygon filling algorithms using the concepts of object oriented programming.

→ **Problem statement**

Write a C++ program to draw a concave polygon and fill it with desired color using scan-fill algorithm. Apply the concept of inheritance.

→ **Outcomes**

i) Understand & apply concept of inheritance for implementing reusable programming codes.

ii) Analyze & apply computer graphics algorithms for line-circle drawing, scan conversion and filling with the help of object-oriented programming concepts.

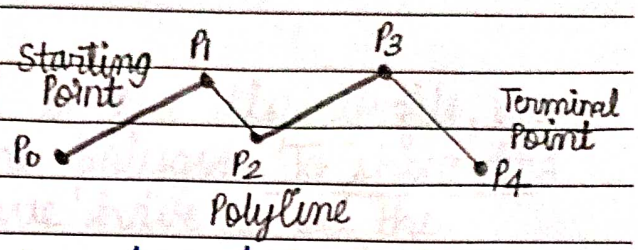
→ **Software & Hardware Requirements**

i) Software : 64-bit open source linux or its derivative
Open source C++, programming tools like G++/GCC, OPENGL

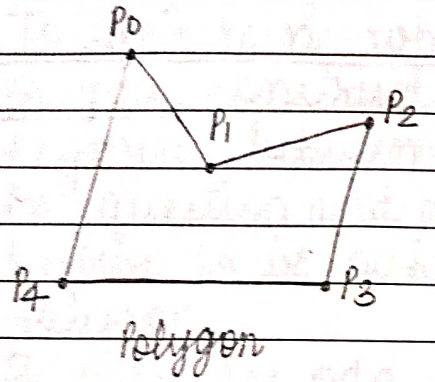
ii) Hardware : Processor → intel (i5)
storage → 1 TB
RAM → 4GB

→ Theory

A polyline is a chain of connected line segments.

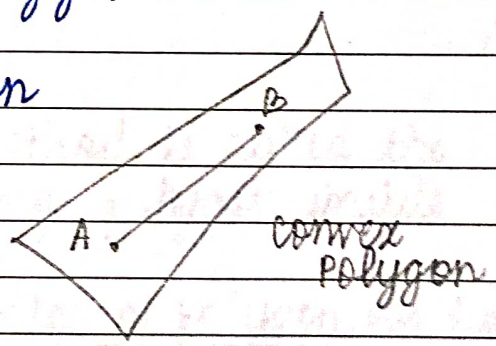


- When the starting and terminal point of a polyline is same, i.e. when polyline is closed, then it is called a polygon. The classification of polygons is based on where the line segment joining any two points within the polygon is going to lie. Thus, there are three types of polygons:-



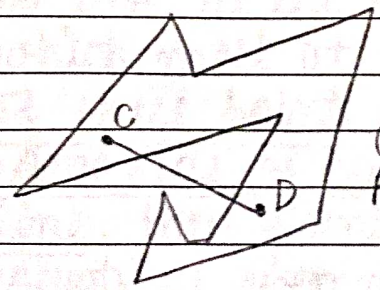
(i) Convex

A convex polygon is a polygon in which the line segment joining any two points within the polygon lies completely inside the polygon.



(ii) Concave

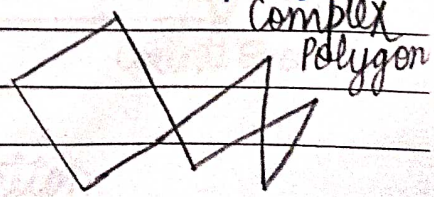
A concave polygon is a polygon in which the line segment joining any two points within the polygon may not lie completely inside the polygon.



Concave Polygon

(iii) Complex

A polygon is simple if it is described by a single non-intersecting boundary, otherwise it is complex.



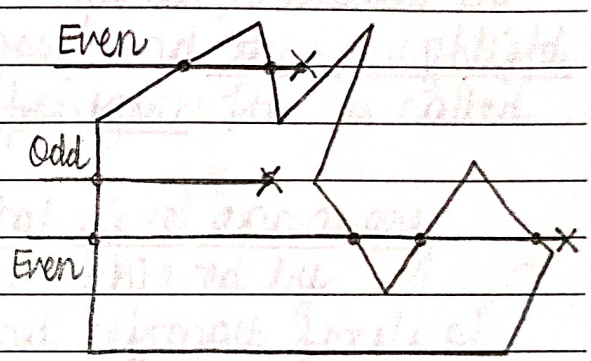
Complex Polygon

→ Inside Test

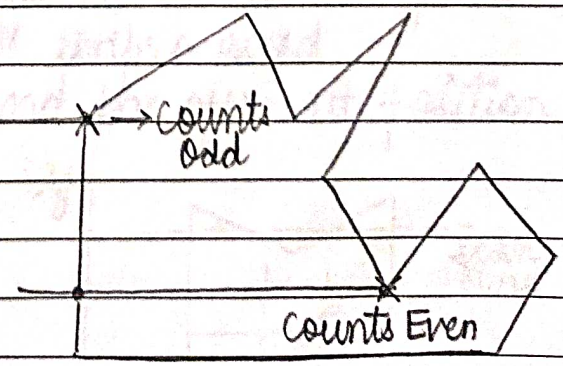
Even-Odd Method

Once the polygon is entered in the display file, we can draw the outline of the polygon. To show the polygon as a solid object, we have to set the pixels inside the polygon as well as pixels on the boundary of it.

- To determine whether a point is inside the polygon or not, construct a line segment between the point in question and a point known to be outside the polygon.



- If there are odd number of intersections, then the point in question is inside, otherwise it is outside. This method is called the even-odd method of determining polygon inside points.
- If the intersection points are vertices of polygon, we have to look at the other two endpoints of the line segments which meet at the vertex.
- If these points lie on the same side of constructed line, then point in question counts as even number of intersections. If they lie on opposite sides of the constructed line, then the point is counted as a single intersection.



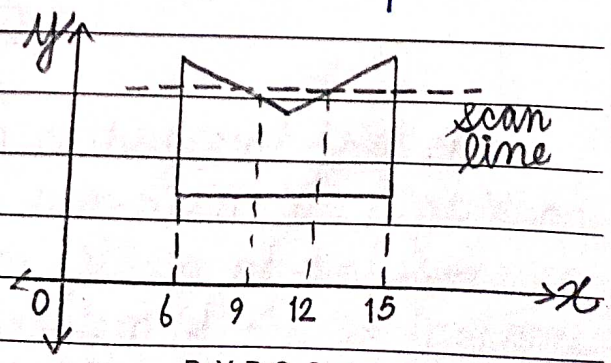
→ Polygon filling

Filling the polygon means highlighting all the pixels which lie inside the polygon with any colour other than the background colour. Polygons are easier to fill since they have linear boundaries.

Scan-line Algorithm

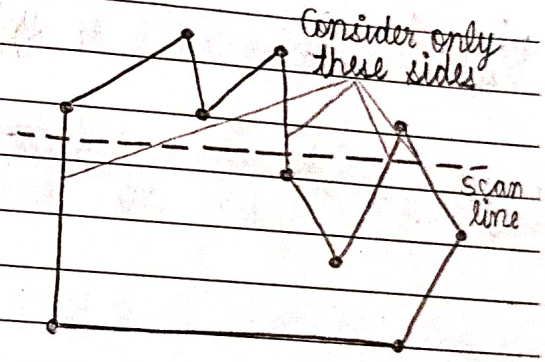
This approach to fill the polygon uses the inside test, i.e. to check whether the pixel is inside the polygon or outside the polygon and then highlight pixels which lie inside the polygon. This is called scan-line algorithm.

- In this method, the horizontal pixel scans are filled across scan lines. This is achieved by identifying the rightmost and leftmost pixels of the seed pixel and then drawing a horizontal line between these two boundary pixels.
- This procedure is repeated with changing the seed pixel above & below the line just drawn until the complete polygon is filled.
- With this efficient method, we have to stack only a beginning position for each horizontal pixel scan, instead of stacking all unprocessed neighbouring positions around the current position.
- For each scan line crossing a polygon, the algorithm locates the intersection points of the scan line with the polygon edges. These intersection points are then sorted from

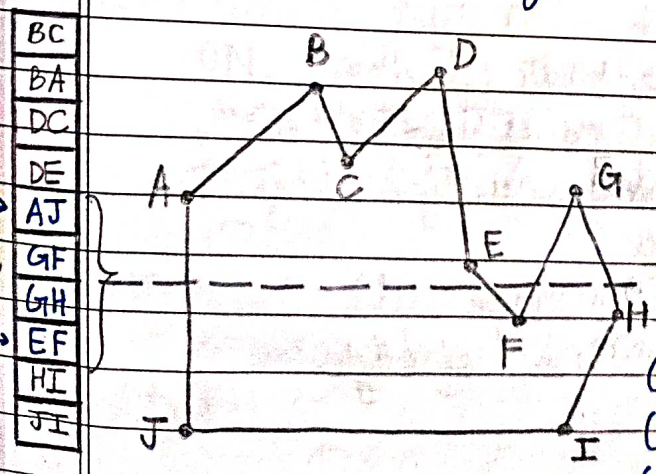


left to right and the corresponding positions between each intersection pair are set to the specified fill colour.

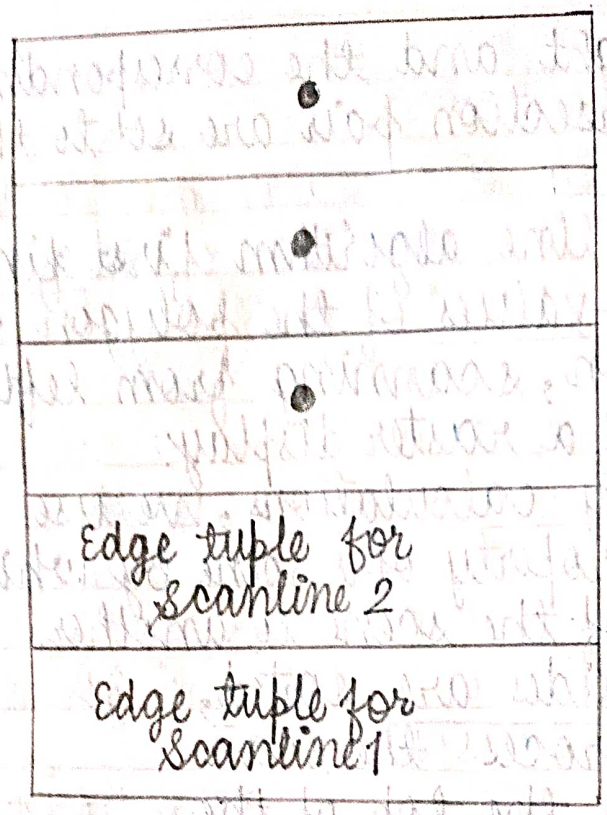
- The scan-line algorithm first finds the smallest & largest y-values of the polygon and works its way down, scanning from left to right, in the manner of a raster display.
- To simplify calculations, we use coherence property. It is the property of a scene by which we can relate one part of the scene to another.
- Once the sides are sorted, we can process the scan lines from the top of the polygon to the bottom, producing an active edge list for each scan line crossing the polygon line.



- The active edge list for a scan-line contains all the edges crossed by that scan line.
- A scan line algorithm for filling a polygon begins by ordering the polygon sides on the largest y values. It begins with largest y-value & works its way down the polygon.



- Features
  (i) Used in orthogonal projections
  (ii) It is a non-recursive algorithm
  (iii) Used for filling of polygons
  (iv) solves problem of hidden surfaces
  (v) We have to stack only beginning position for each pixel scan.



EDGE TUPLE

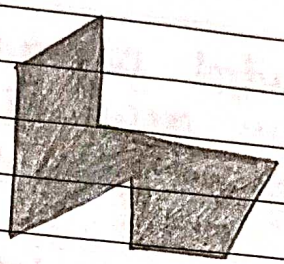
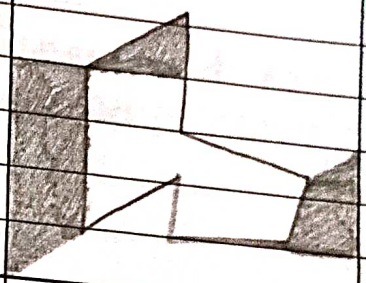
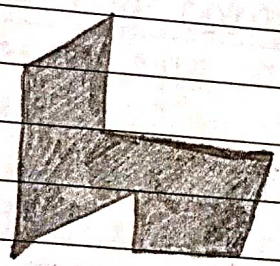
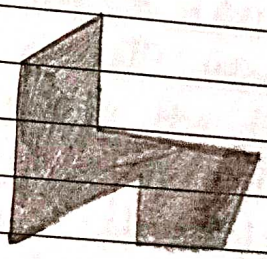
y-min	y-max	x-value of y-min	Slope Inverse

y-max : maximum y-coordinate of edge
x of y-min : x-coordinate of lowest edge point, updated at every scanline while scanline filling
Slope Inverse : inverse of edge slope
y-min : minimum y-coordinate of edge

→ Algorithm

1) Start
2) Initialize the derived data structure -
   (i) Create a polygon table having colour, edge, pointers, coefficients.
   (ii) Establish edge table which contains information regarding the endpoint of edges, pointer to polygon and inverse slope.
   (iii) Create active edge list. This will be sorted in increasing order of x.
   (iv) Create a flag 'F'. It will have two values, either on or off.
3) Perform the following steps for all scan lines
   (i) Enter values in Active Edge List (AEL) in sorted order using 'y' as value.
   (ii) Scan, until the flag, i.e. 'F' is on, using a background colour.
   (iii) When one polygon flag is ON, and this is for surface $S1$, enter colour intensity as $I1$ into the refresh buffer.
   (iv) When two or more image surface flags are ON, sort the surfaces according to depth and use intensity values I_n for the nth surface. This surface will have the least Z depth value.
   (V) Use the concept of coherence for the remaining planes
4) Stop

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Print the polygon			Fail
2.	Print the polygon			Pass

→ Conclusion

Hence, we have succesfully implemented scanline polygon fill algorithm using object oriented programming concepts

21/10/22

OOP
Assignment - A3

→ **Title**

To develop a program that instantiates the book and tape class using various functions of C++.

→ **Objectives**

To learn the concepts inheritance and exception handling, and to learn various other concepts of object oriented programming.

→ **Problem statement**

Imagine a publishing company which does marketing for book and audio cassette versions. Create a class publication that stores the title (a string) and price (type float) of publications. From this class, derive two classes: book which adds a page count (type int) and tape which adds a playing time in minutes (type float).

Write a program that instantiates the book and tape class, allows user to enter data and displays the data members. If an exception is caught, replace all the data members' values with zero values.

→ **Outcomes**

Understand and apply the concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

→ Software & Hardware Requirements

i) Software : Ubuntu/Linux (OS), Open source C++ Programming tools like G++/GCC.

ii) Hardware : Processor → intel (i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

Inheritance

Inheritance is a property in which data members and member functions of some class are used by some other class.

- Inheritance allows the reusability of code in C++
- The class from which the data members and member functions are used by another class is called the base class (or parent class or superclass)
- The class which uses the properties of base class and at the same time can add its own properties is called derived class (or subclass or child class)

→ Access specifiers

i) Private

If a base class has private members, then those members are not accessible to the derived class.

ii) Protected

Protected members are public to the derived class but private to rest of the program.

iii) Public

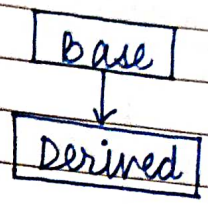
Public members are accessible to all.

- The derived class can inherit the base class publically or privately.

Types of Inheritance

I. Single inheritance

In single inheritance, there is one parent class per derived class.

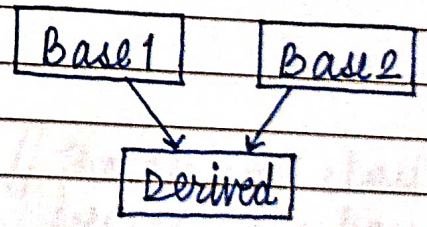


- **Syntax**

```
class parent
    {
        visibility mode:
            data member(s);
            member function(s);
    };
class child: public parent
    {
        ...
    };
```

II. Multiple Inheritance

In multiple inheritance, the derived class is derived from more than one base class.

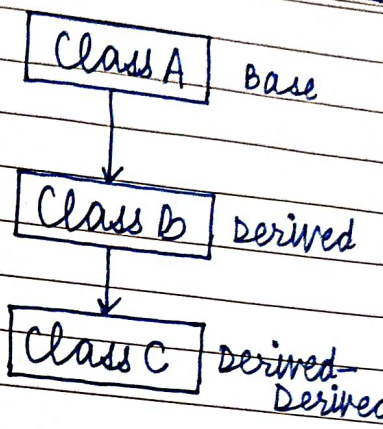


- **Syntax**

```
class base1 {
        ...
    };
class base2 {
        ...
    };
class derived: public base1, public base2 {
        ...
    };
```

III. Multilevel Inheritance

When a derived class is derived from a base class which itself is a derived class then that type of inheritance is called multilevel inheritance.

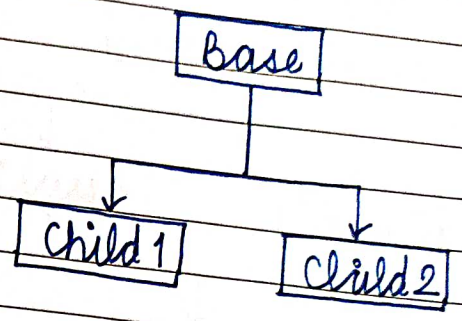


- Syntax

```
class parent
    {
        visibility mode:
        data member(s);
        member function(s);
    };
class derived1 : public parent
    {
        visibility mode:
        data member(s);
        member function(s);
    };
class derived2 : public derived1    // The derived1 class
    {                                becomes the base
        visibility mode:             class for the
        data member(s);              derived2 class.
        member function(s);
    };
class derived3 : public derived2
    {
        :
        :
```

IV. Hierarchical Inheritance

Hierarchical inheritance is a kind of inheritance in which one or more classes are derived from the common base class.

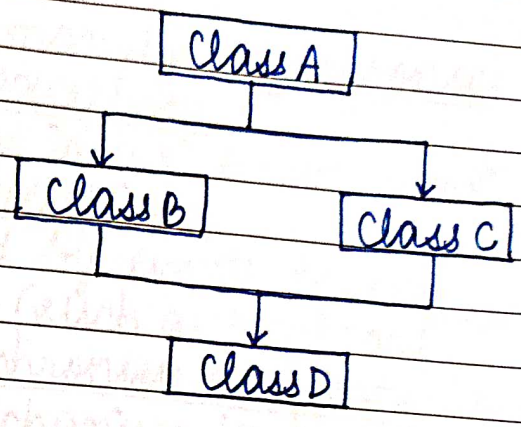


- Syntax

```
class base
    {
      visibility mode:
      data member(s);
      member function(s);
    };
class derived1: protected base
    {
      visibility mode:
      data member(s);
      member function(s);
    };
class derived 2: protected base
    {
      visibility mode:
      data member(s);
      member function(s);
    };
class derived 3: protected base
    {
      :
      :
```

V. Hybrid Inheritance

When two or more types of inheritances are combined together then it forms the hybrid inheritance.



- In the diagram, the path A→B shows single inheritance, the path A→B, A→C shows hierarchical inheritance, the path B→D, C→D shows multiple inheritance and A→B→D shows multilevel inheritance.
- Syntax

```
class parent
    {
        visibility mode:
        ...
    };
class child1 : public parent {
        ...
    };
class child2 : public parent {
        ...
    };
class derived : public child1, public child2
    {
    visibility mode:
    data member(s);
    member function(s);
    };
```

Exception Handling

Exceptions are of two kinds, namely, synchronous exceptions & asynchronous exceptions.

- Errors such as 'out of range index' & 'over-flow' belong to the synchronous exceptions.
- The errors that are caused by events beyond the control of the program (such as keyboard interrupts) are called asynchronous exceptions. The exception handling mechanism in C++ is designed to handle only the synchronous exceptions.
- The purpose of exception handling mechanism is to provide means to detect & report an 'exceptional circumstance' so that appropriate action can be taken. The mechanism suggests a separate error handling code that performs the following tasks-

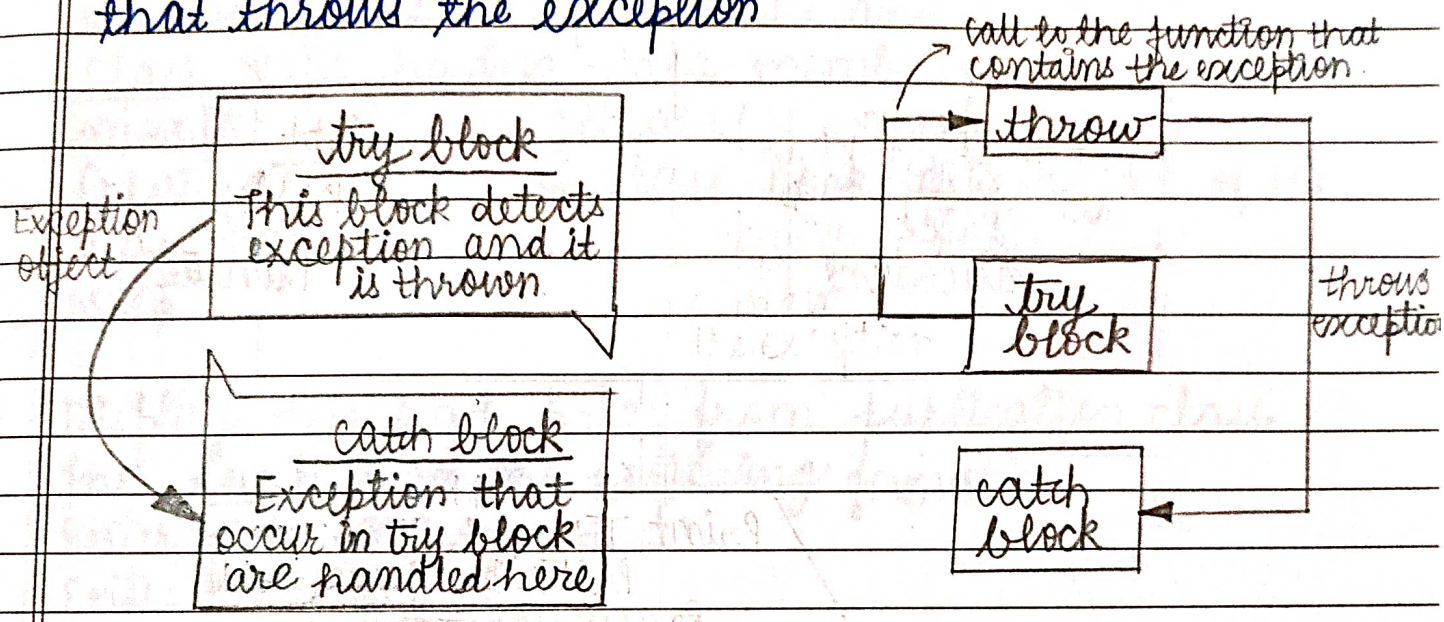
i) Find the problem (Hit the exception)
ii) Inform that an error has occured (Throw the exception)
iii) Recieve the error information (Catch the exception)
iv) Take corrective actions (Handle the exception)

→ The mechanism

C++ exception handling mechanism is basically built upon three keywords; 'try', 'throw' and 'catch'

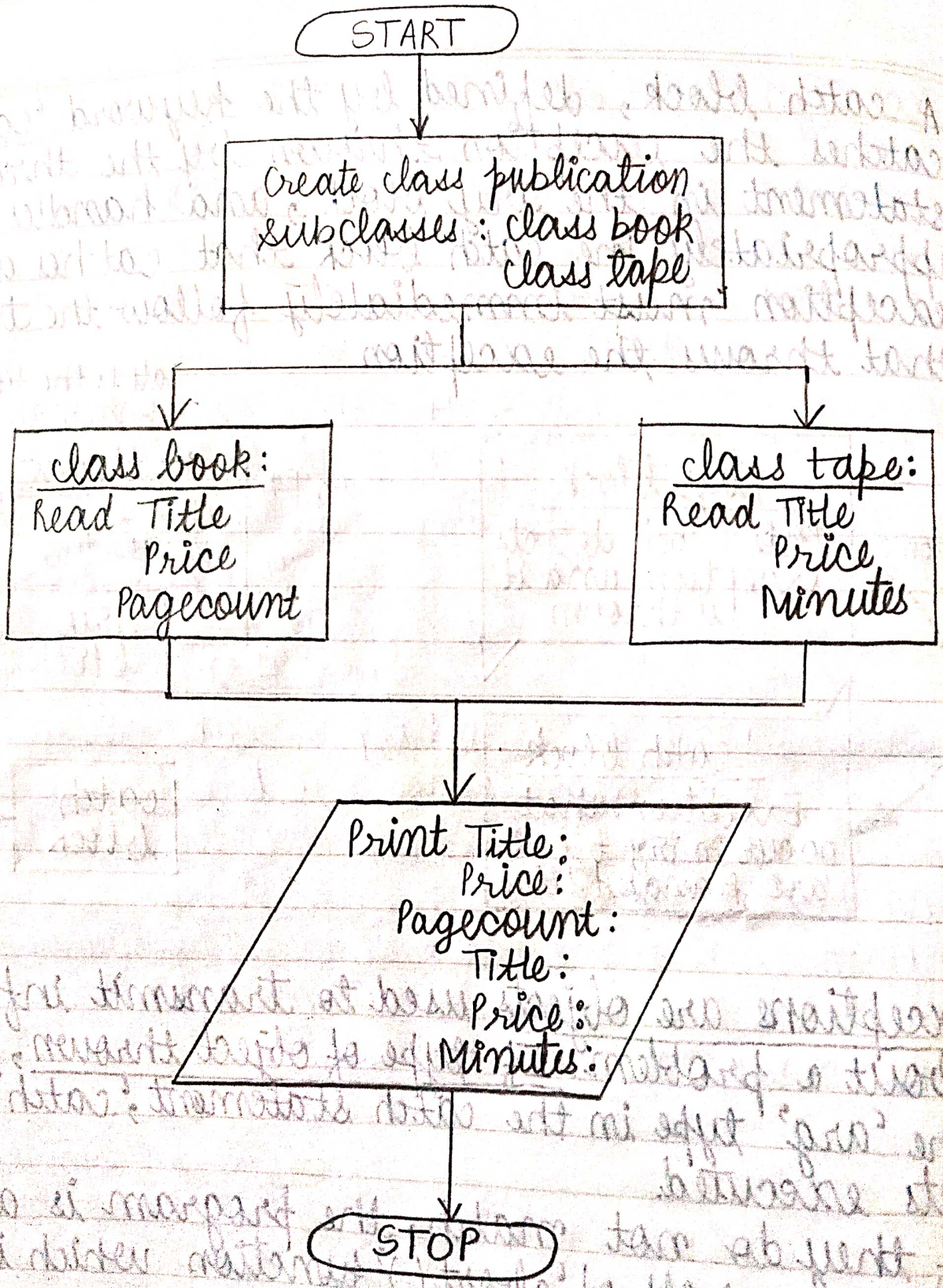
- The keyword 'try' is used to preface a block of statements, surround by braces, which may generate exceptions. This block of statements is known as try block.
- When an exception is detected, it is used to 'throw' the statement in the 'try' block.

- A catch block, defined by the keyword 'catch' catches the exception thrown by the throw statement in the try block, and handles it appropriately. The catch block that catches an exception must immediately follow the try block that throws the exception



- Exceptions are objects used to transmit information about a problem. If type of object thrown, matches the 'arg' type in the catch statement, 'catch' block gets executed.
If they do not match, the program is aborted with the help of 'abort()' function which is invoked by default
When no exception is detected and thrown, the control goes to the statement immediately after the catch block, i.e., the catch block is skipped.

→ Flowchart


START
Create class publication
subclasses : class book
class tape
class book:
Read Title
Price
Pagecount
class tape:
Read Title
Price
Minutes
Print Title:
Price:
Pagecount:
Title:
Price:
Minutes:
STOP

→ Algorithm

i) Start
ii) Create classes publication, book and tape.
iii) Publication class having data members title & price
iv) Class book having data members pages and member functions getdata() and putdata().
v) Class tape having data members minutes and member functions getdata() and putdata()
vi) Create an object of class book and object of the class tape.
vii) Stop

- Display title and price from publication class. The result is in the following format:-

Enter title: ODP
Enter Price: 350
Enter Pages: 450
Enter title: CG
Enter Price: 200
Enter minutes: 45

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Accept the information from the user.	Enter Title: Enter Price: Enter Pages: ⋮	Program finished with exit code 0	Fail
2.	Accept the information from the user.	Enter Title: Enter Price: Enter Pages: Enter Title: Enter Price: Enter Minutes:	Same as the expected output.	Pass
3.	Display the information provided by the user.	Title: DDP Price: 350 Pagecount: 450 Title: CG Price: 200 Minutes: 45	Same as the expected output.	Pass
4.	Exit	End	End	Pass

→ Conclusion

Hence, we have successfully learnt the concepts of inheritance & exception handling using object oriented programming.

21/10/22

CG Assignment-A2

→ **Title**

To apply Cohen-Sutherland line clipping algorithm.

→ **Objective**

To understand basics of computer graphics, apply various methods & techniques for implementing line clipping algorithms using the concepts of object oriented programming.

→ **Problem Statement**

Write a C++ program to implement Cohen-Sutherland line clipping algorithm

→ **Outcomes**

i) Understand the concepts of inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

ii) Understand the concept of windowing and clipping and apply various algorithms to fill and clip lines & polygons.

→ **Software & Hardware Requirements**

i) **Software**: 64-bit open source linux or its derivative, Open source C++, programming tools like G++/GCC, OPENGL

ii) **Hardware**: Processor → Intel (i5)
Storage → 1 TB
RAM → 4 GB

1001	1000	1010
0001	WINDOW 0000	0010
0101	0100	0110

Four-bit codes for the NINE REGIONS

→ Theory

Cohen-Sutherland Line Clipping Algorithm

Cohen-Sutherland algorithm is one of the oldest and most popular line clipping algorithm developed by Dan Cohen & Ivan Sutherland.

- To speed up the processing, this algorithm performs initial tests that reduce the number of intersections that must be calculated.
- This algorithm uses a four digit (bit) code to indicate which of the nine regions contains the end point of line.
- The four-bit codes are called region codes or outcodes. These codes identify the location of the point relative to the boundaries of the clipping rectangle.
- Each bit position in the region code is used to indicate one of the four relative co-ordinate positions of the point with respect to the clipping window: to the left, right, top or bottom.

(i) Set bit1
- if the end point is to the left of the window.

(ii) Set bit2
- if the end point is to the right of the window

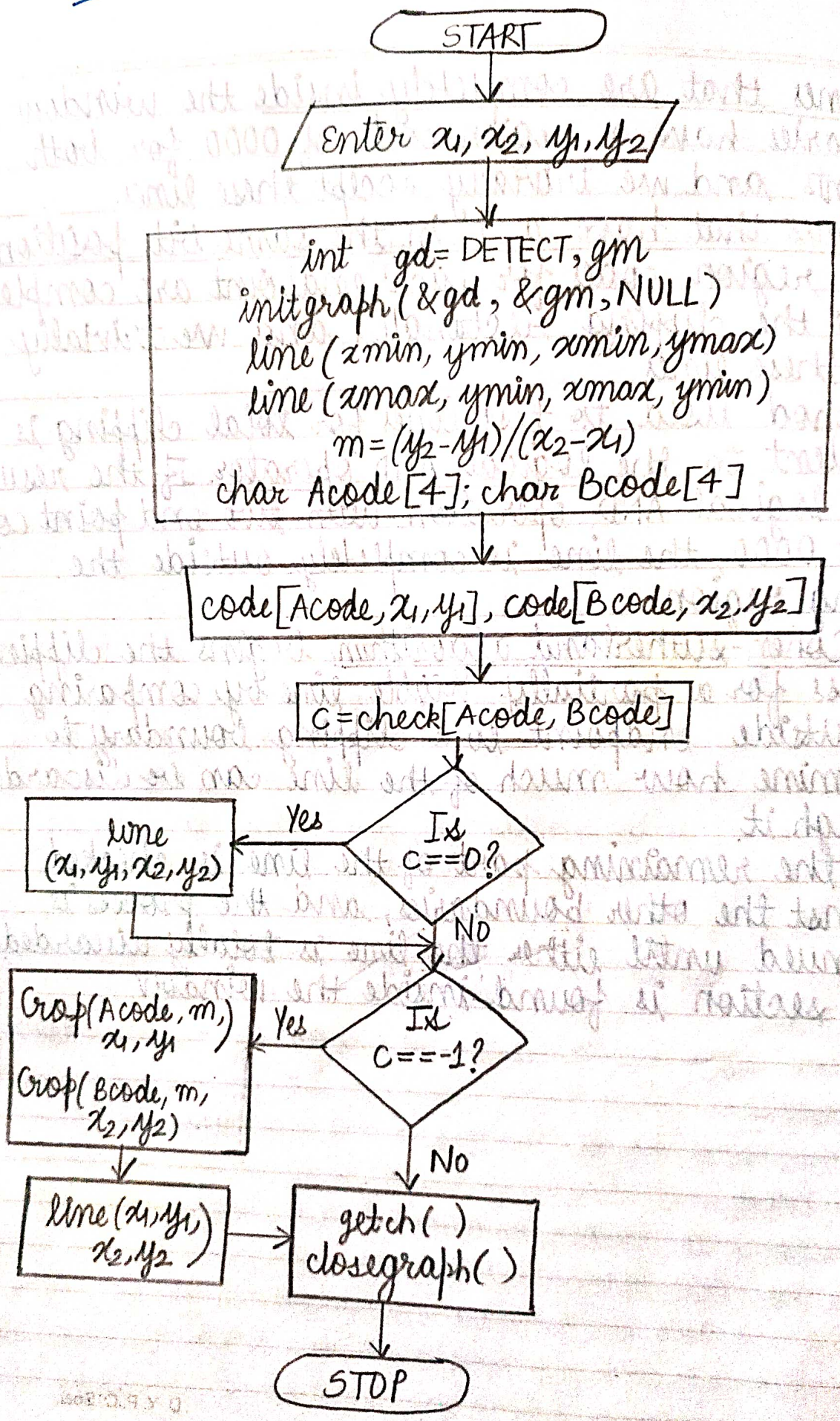
(iii) Set bit3
- if the end point is below the window.

(iv) Set bit4
- if the end point is above the window.

- Otherwise, the bit is set to zero.
- Once we have established region codes for all line endpoints, we can determine which lines are completely inside the clipping window & which are clearly outside

- Any lines that are completely inside the window boundaries have a region code of 0000 for both endpoints and we trivially accept these lines.
- Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping rectangle, and we trivially reject these lines
- A method used to test lines for total clipping is equivalent to the logical AND operator. If the result of the logical AND operation with two end point codes is not 0000, the line is completely outside the clipping region.
- The Cohen-Sutherland algorithm begins the clipping process for a partially visible line by comparing an outside endpoint to a clipping boundary to determine how much of the line can be discarded through it.
- Then the remaining part of the line is checked against the other boundaries, and the process is continued until either the line is totally discarded or a section is found inside the window.

→ Flowchart


START
Enter x1, x2, y1, y2
int gd= DETECT, gm
initgraph(&gd, &gm, NULL)
line(xmin, ymin, xmin, ymax)
line(xmax, ymin, xmax, ymin)
m=(y2-y1)/(x2-x1)
char Acode[4]; char Bcode[4]
code[Acode, x1, y1], code[Bcode, x2, y2]
C=check[Acode, Bcode]
Is C==0?
Yes
line (x1, y1, x2, y2)
No
Is C==-1?
Yes
Crop(Acode, m, x1, y1)
Crop(Bcode, m, x2, y2)
No
line(x1, y1, x2, y2)
getch()
closegraph()
STOP

→ Algorithm

1. START
2. Read two end points of the line, say $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$
3. Read two corners (left-top and right-bottom) of the window, say $(Wx_1, Wy_1$ and $Wx_2, Wy_2)$
4. Assign the region codes for two endpoints P_1 & P_2 using following step. Initialize code with bits 0000.
   (i) set bit 1 – if $(x < Wx_1)$
   (ii) set bit 2 – if $(x > Wx_2)$
   (iii) set bit 3 – if $(y < Wy_2)$
   (iv) set bit 4 – if $(y > Wy_1)$
5. Check for visibility of line P_1P_2
   (i) If region codes for both P_1 & P_2 are zero, then line is completely visible
   (ii) If region codes are not zero and logical ANDing is also non-zero, line is completely invisible.
   (iii) If region codes don't satisfy 5(i) & 5(ii), line is partially visible
6. Determine the intersecting edge of clipping window by inspecting the region codes of two endpoints
   (i) If both P_1 & P_2 are non-zero, find intersection points P_1' & P_2'
   (ii) If region code for only one is zero, then find intersection point for the one which is non-zero P_1' or P_2'.
7. Divide the line segments considering intersection points
8. Reject the line segment if any one end point of it appears outsides the clipping window.
9. Draw the remaining line segments.
10. STOP

→ Test Cases

S. no.	Input	Expected Output	Actual Output	Result
1.				Pass
2.				Fail
3.				Pass
4.				Fail
5.				Pass

→ Conclusion

Thus, we successfully implemented the Cohen-Sutherland line-clipping algorithm.

6/12

CG
Assignment - B4

→ **Title**

To use concepts of 2D Transformations like Scaling, Translation & Rotation.

→ **Objective**

To understand basics of computer graphics, apply various methods & techniques for implementing 2D Transformations using Object Oriented Programming

→ **Problem statement**

Write a C++ program to draw 2-D objects and perform basic transformations, like

i) Scaling
ii) Translation
iii) Rotation

→ **Outcomes**

To understand the concepts of basic transformations on a two-dimensional object by using object oriented programming

→ **Software & Hardware Requirements**

i) **Software** : 64-bit open source linux or its derivative, Open source C++, programming tools like G++/GCC, OPENGL

ii) **Hardware** : Processor → intel (i5)
Storage → 1 TB
RAM → 4GB

→ Example

eg. Translate a polygon ABCD with co-ordinates A(2,5); B(4,5); C(4,3); D(2,3) if T = (1,5).

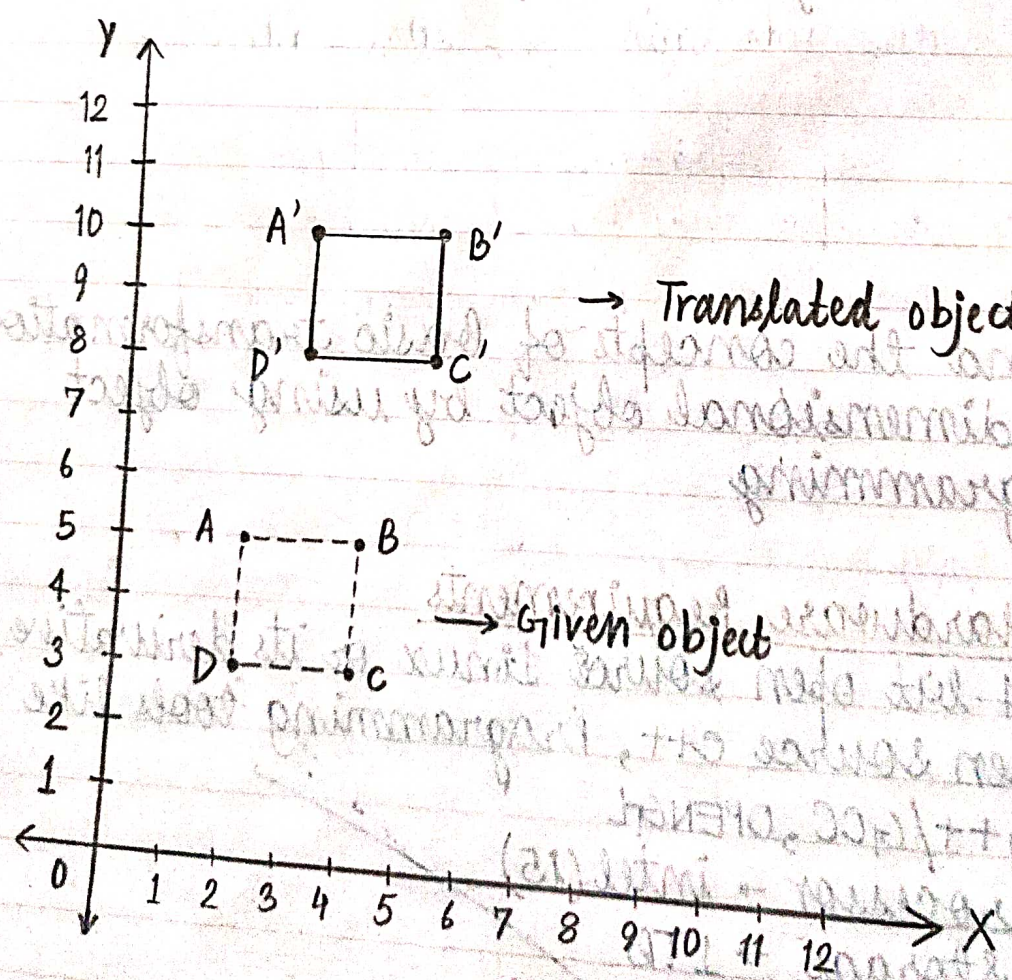
Ans

$$A' = A + T = \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 \\ 10 \end{bmatrix}$$

$$B' = B + T = \begin{bmatrix} 4 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \end{bmatrix}$$

$$C' = C + T = \begin{bmatrix} 4 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

$$D' = D + T = \begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$



Translation

→ Theory

I. Translation

Translation is a process of changing the position of an object in a straight-line path from one co-ordinate location to another.

- We can translate a two-dimensional point by adding translation distances t_x, t_y to the original co-ordinate positions (x, y) to move the point to a new position (x', y')

$$\boxed{\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}}$$

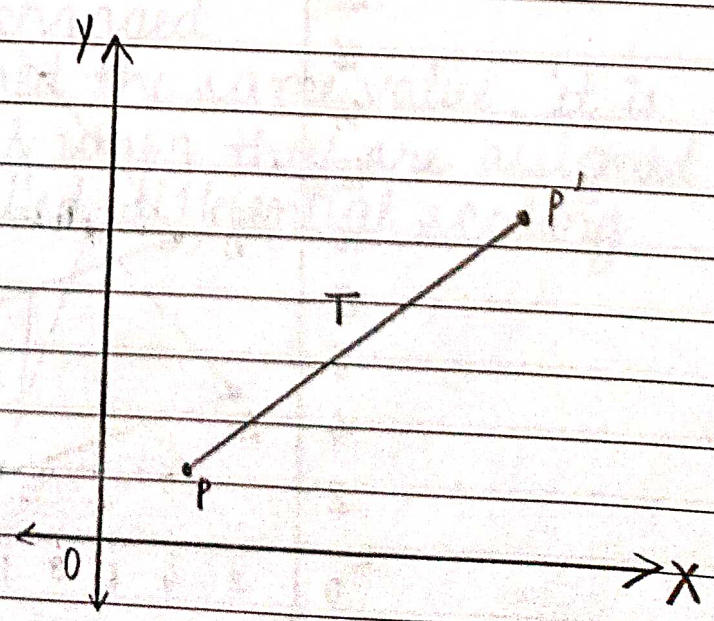
The translation distance pair (t_x, t_y) is called a translation vector or shift vector.

We can express the translation equations as a single matrix equation by using column vectors to represent co-ordinate positions and the translation vector:

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\boxed{P' = P + T}$$



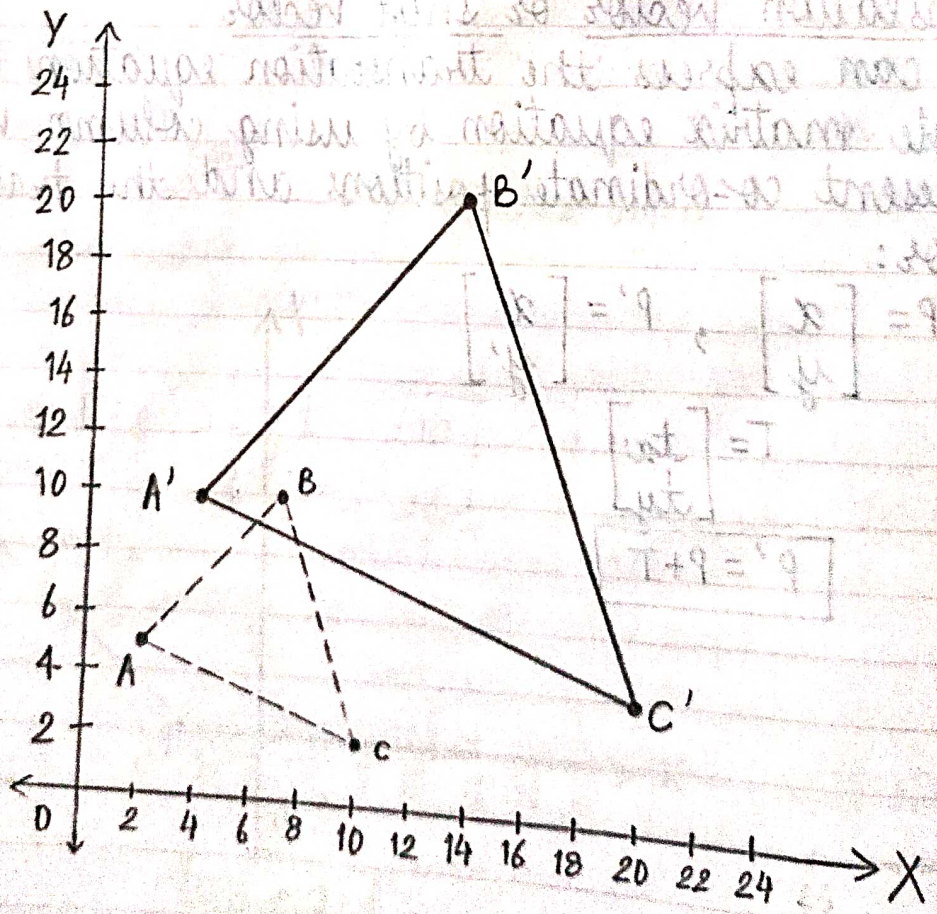
→ Example

eg. scale the polygon with co-ordinates A(2,5), B(7,10), C(10,2) if S=[2,2]

Ans As $S=[2,2]$, thus
$S_x = 2$ & $S_y = 2$

Object matrix :
$$\begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix} \end{array}$$

$$\begin{array}{c} A' \\ B' \\ C' \end{array}\begin{bmatrix} x_1' & y_1' \\ x_2' & y_2' \\ x_3' & y_3' \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 7 & 10 \\ 10 & 2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 10 \\ 14 & 20 \\ 20 & 4 \end{bmatrix}$$



Scaling

II. Scaling

A scaling operation changes the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors 'sx' & 'sy' to produce the transformed coordinates (x', y')

$$\boxed{\begin{aligned} x' &= x * Sx \\ y' &= y * Sy \end{aligned}}$$

- Scaling factor 'sx' scales the objects in x-direction while 'sy' scales them in the y-direction
- Any positive numeric value can be assigned to scaling factors 'sx', 'sy'.

(i) If value < 1,
reduce the size of the object

(ii) If value > 1,
enlarge the object

(iii) If value $= 1$,
the object remains unchanged

- When 'sx', 'sy' are assigned the same value, it is called uniform scaling & when they are assigned unequal values, it is called differential scaling.

$$[x', y'] = [x \quad y] \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

$$[x', y'] = [x \cdot Sx \quad y \cdot Sx]$$
$$= P \cdot S$$

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

But for clockwise direction,

$$R = \begin{bmatrix} \cos(-\theta) & \sin(-\theta) \\ -\sin(-\theta) & \cos(-\theta) \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

eg. A point (4, 3) is rotated counterclockwise by an angle of 45°. Find the rotation matrix and find the resultant point

Ans $R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \cos 45^\circ & \sin 45^\circ \\ -\sin 45^\circ & \cos 45^\circ \end{bmatrix}$

$$R = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$P' = \begin{bmatrix} 4 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 4/\sqrt{2} - 3/\sqrt{2} & 4/\sqrt{2} + 3/\sqrt{2} \end{bmatrix}$$

$$P' = \begin{bmatrix} 1/\sqrt{2} & 7/\sqrt{2} \end{bmatrix}$$

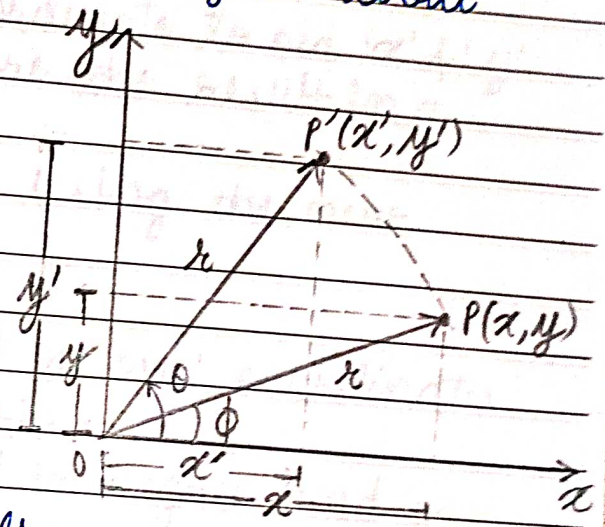
EXAMPLE

Rotation

III. Rotation

A two dimensional rotation is applied to an object by repositioning it along a circular path in the x-y plane.

- Points can be rotated along an angle 'θ' about the origin.
- Here, 'r' is the constant distance of the point from the origin, angle 'ϕ' is the original angular position of the point from the horizontal & angle 'θ' is the rotation angle.



Counterclockwise rotations are defined by positive values of rotation angle (+θ) & clockwise rotations are defined by negative values of angle of rotation (-θ).

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \quad \ldots (1)$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \quad \ldots (2)$$

The original co-ordinates of points are-

$x = r\cos\phi$, $y = r\sin\phi$

substituting in equations (1) & (2),

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

In matrix form,

$$[x' \;\; y'] = [x \;\; y]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$P' = P.R$

; where R = rotation matrix

→ Algorithm

1) START

For Translation

2) Take input from the user for original x's & y's of the object coordinates.
3) Add 'x' & 'y' translation coordinate to old 'x' & 'y' coordinates respectively. Store the result in a new matrix
4) Plot the transformed figure using the new matrix

For rotation

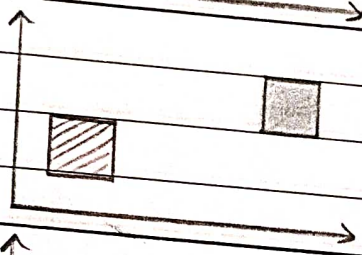
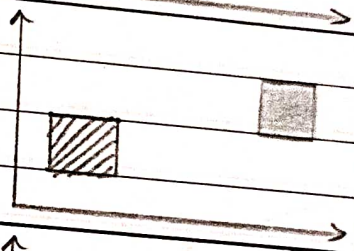
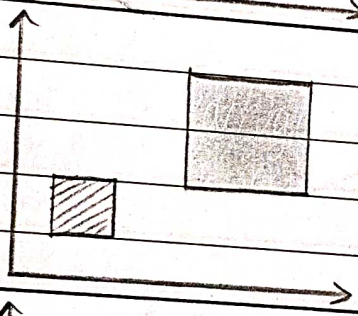
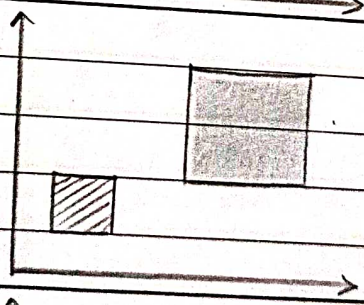
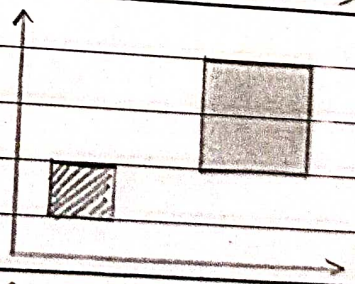
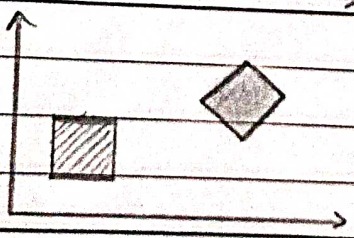
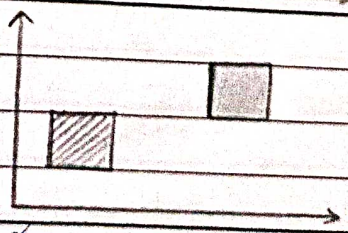
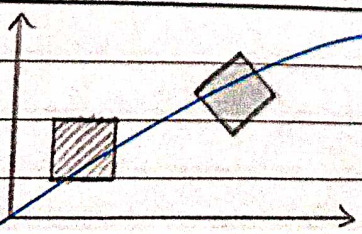
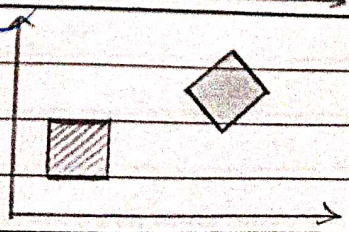
5) Take user input of original 'x' & 'y' coordinates and store them in a matrix.
6) Take user input of angle 'θ' in degree and compute the rotation matrix as-

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For scaling

7) Take input from the user for x's & y's of the original object.
8) Take input from the user for the scaling factors in x & y directions. Multiply it by the old co-ordinates.
9) Plot the transformed object.
10) STOP

→ Test Cases

S. No.	Step	Expected Output	Actual Output	Result
1.	Translation			Fail
2.	Translation			Pass
3.	Scaling			Fail
4.	Scaling			Pass
5.	Rotation			Fail
6.	Rotation			Pass

→ Conclusion

Thus, we have successfully created a C++ program to perform transformations like scaling, translation and rotation on a 2D object.

OOP
Assignment – A1

→ **Title**

To perform various operations on complex numbers like addition, multiplication using operator overloading.

→ **Objectives**

To understand & apply concepts like operator overloading, polymorphism, inheritance for implementing reusable programming codes using object oriented programming.

→ **Problem statement**

Implement a class Complex which represents the Complex Number data type. Implement the following -

1. Constructor (including a default constructor which creates the complex number 0 + 0i)
2. overload operator + to add two complex numbers.
3. Overload operator * to multiply two complex numbers.
4. overload operators << and >> to print and read complex numbers.

→ **Outcomes**

Understand & apply various object oriented programming concepts for implementing reusable program codes.

→ **Software & Hardware Requirements**

i) Software : 64-bit open source linux or its derivative open source C++, G++/GCC tools, OPENGL

ii) Hardware : Processor → intel(i5), storage → 1TB, RAM → 4GB

→ Theory

Class

A class is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of the class.

- A C++ class is like a blueprint for an object. A class contains data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members & member functions define the properties & behaviour of the objects in a class.
- We need to define access specifiers when creating a class. They can be private, protected or public. In case, none is mentioned, it gets private mode by default.

→ Syntax

```
class name_of_class
    {
    private:
        variable(s);
        function(s);
    public:
        variable(s);
        function(s);
    };
```

Operator Overloading

Operator overloading is compile-time polymorphism. It is an idea of giving special meaning to an existing operator in C++ without changing its meaning.

→ syntax

```
Return type :: operator symbol (Arguments list)
{
  // function body
}
```

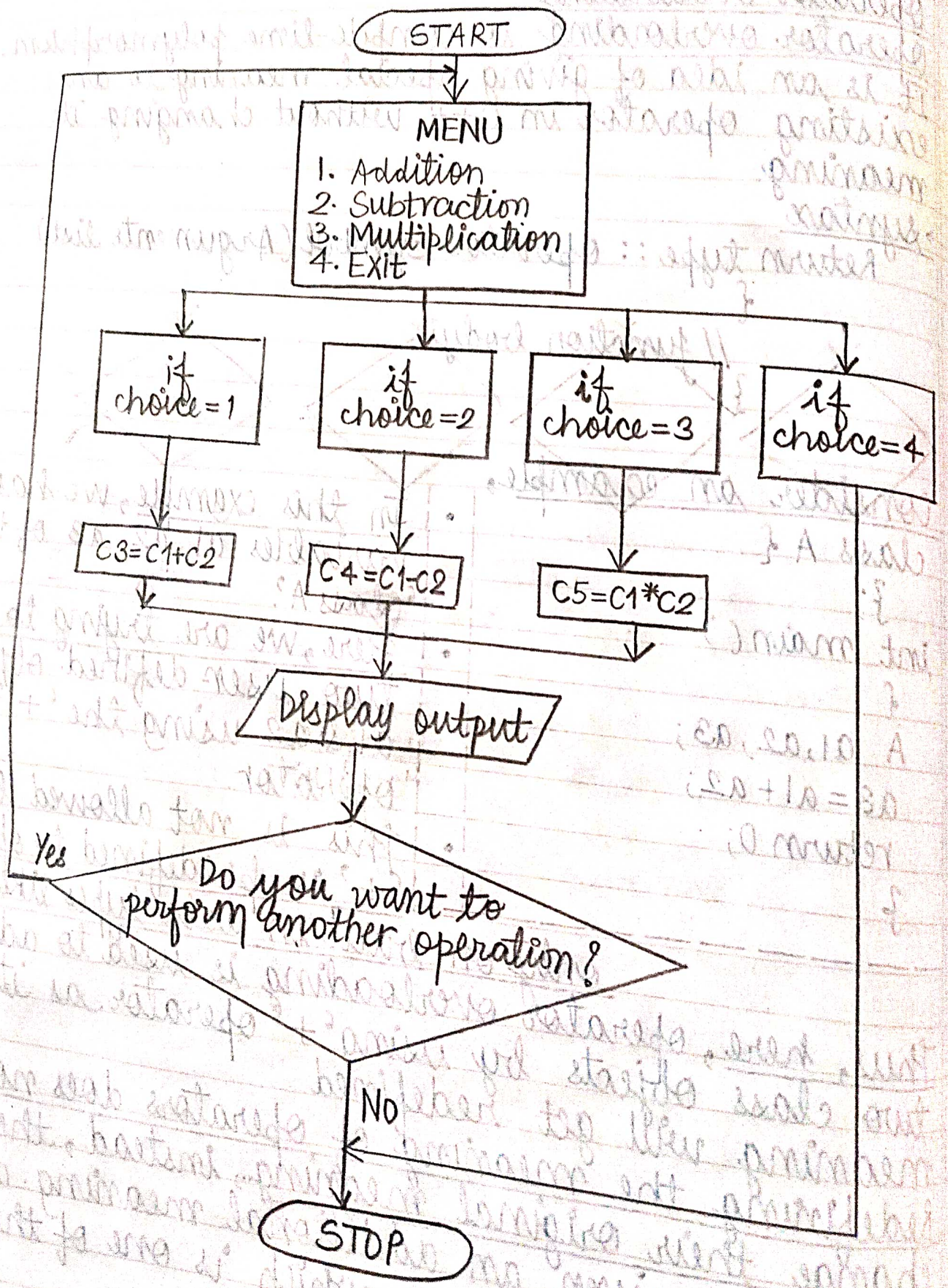
- Consider an example,

```
class A {
  };
int main()
  {
  A a1,a2,a3;
  a3=a1+a2;
  return 0;
  }
```

- In this example, we have variables a1, a2, a3 of type 'class A'.
- Here, we are trying to add two user-defined objects a1 & a2 using the '+' operator.
- This is not allowed because '+' is predefined to operate only on built-in data types like int.

- Thus, here, operator overloading is used to add two class objects by using '+' operator as its meaning will get redefined.
- Redefining the meaning of operators does not change their original meaning, instead, they have been given an additional meaning along with their existing ones which is one of the basic foundations of polymorphism.

→ Flowchart


START
MENU
1. Addition
2. Subtraction
3. Multiplication
4. Exit
if choice=1
if choice=2
if choice=3
if choice=4
C3=C1+C2
C4=C1-C2
C5=C1*C2
Display output
Do you want to perform another operation?
Yes
No
STOP

→ Algorithm

1. START
2. Create a class Complex and make five objects of this class.
3. Input C1, C2 using the overloaded '>>' (extraction) operators.
4. Set C3 = C1+C2 using the overloaded '+' (addition) operators as normal '+' operator can't add two user defined data types.
5. Print C3 using overloaded '<<' (insertion) operator.
6. Set C4 = C1-C2 using the overloaded '–' (subtraction) operator as normal '–' operator can't subtract two user-defined data types.
7. Print C4 using overloaded '<<' (insertion) operator
8. Set C5 = C1*C2 using the overloaded '*' (multiplication) operator as normal '*' operator can't multiply two user defined data types
9. Print C5 using overloaded '<<' (insertion) operator.
10. STOP

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Addition of Complex numbers.	C1 = 1+1i ↵ C2 = 2+2i ↵ C3 = 3+3i	C1 = 1+1i ↵ C2 = 2+2i ↵ C3 = 0+0i	Fail
2.	Addition of complex nos.	C1 = 5+5i ↵ C2 = 7+7i ↵ C3 = 12+12i	C1 = 5+5i ↵ C2 = 7+7i ↵ C3 = 12+12i	Pass
3.	Subtraction of complex numbers	C1 = 3+3i ↵ C2 = 2+2i ↵ C4 = 1+1i	C1 = 3+3i ↵ C2 = 2+2i ↵ C4 = 1+1i	Pass
4.	Multiplication of complex numbers	C1 = 1+1i ↵ C2 = 1+1i ↵ C5 = 0+2i	C1 = 1+1i ↵ C2 = 1+1i ↵ C5 = 2+2i	Fail
5.	Multiplication of complex numbers	C1 = 1+1i ↵ C2 = 1+1i ↵ C5 = 0+2i	C1 = 1+1i ↵ C2 = 1+1i ↵ C5 = 0+2i	Pass

→ Conclusion

Thus, complex numbers were successfully added & subtracted using basic concepts of C++ language.

OOP Assignment - B4

→ **Title**

To perform various operations on files like opening & closing them, reading & writing, etc.

→ **Objectives**

To understand the concepts of file handling using concepts of object oriented programming like polymorphism, inheritance, etc.

→ **Problem statement**

Write a C++ program that creates an output file, writes information in it, closes the file, opens it again as an input file and read information from the file.

→ **Outcomes**

i) Understand and apply concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

ii) Analyze the concept of file and apply it while storing & retrieving the data from secondary storages.

→ **Software & Hardware Requirements**

i) **Software**: 64-bit open source Linux or its derivative, Open source C++, Programming tools like G++/GCC, OPENGL

ii) **Hardware**: Processor → intel (i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

C++ provides following classes to perform input and output of characters to & from the files.

(i) ofstream : This stream class is used to write in files.
(ii) ifstream : This stream class is used to read from files.
(iii) fstream : This stream class is used for both read and write operations from/to files.

→ Opening File Operation

The file can be opened by the function called open().
The syntax of 'file open' is -

```
Open(file-name,mode)
```

- The filename is a null-terminated string that represents the name of the file that is to be opened.
  - ios::in - open for input operation
  - ios::out - open for output operation
  - ios::app - for appending the output operations
- The isopen() is a boolean function that can be used to check whether the file is open or not.

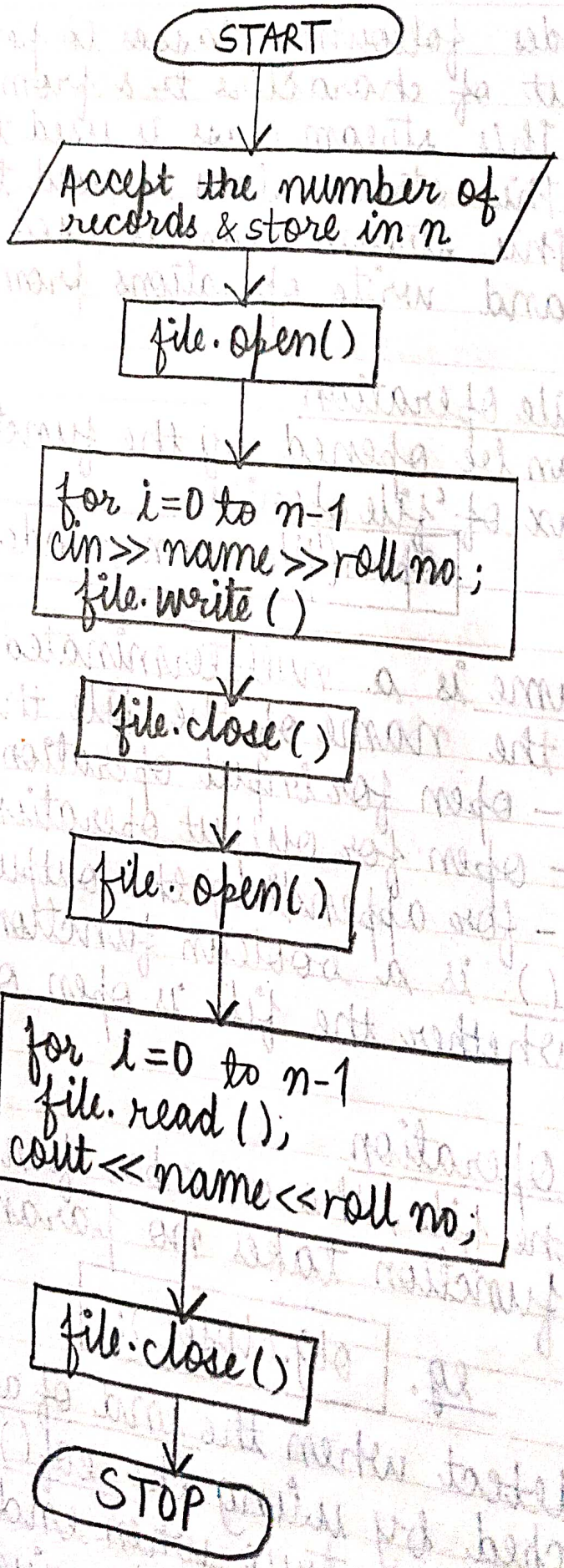
→ Close File Operation

To close the file, the member function close() is used. The close function takes no parameter and returns no value.

eg.
```
obj.close();
```

- You can detect when the end of an input file has been reached by using the eof() member function of ios. It returns true when end of file has been encountered and false otherwise

→ Flowchart


START
Accept the number of records & store in n
file.open()
for i=0 to n-1
cin>> name >>roll no ;
file.write ()
file.close()
file.open()
for i=0 to n-1
file.read ();
cout<< name<<roll no;
file.close()
STOP

→ **Algorithm**

1. START
2. Create a class and define data members, roll numbers and name.
3. Define accept() to take name & roll number from the user.
4. Define display() function to display the record.
5. Create object of class & fstream class.
6. Take a limit for user of number of data records.
7. Open the file in out mode and call accept() to take record from user, then call write() to write that record into the file and at the end, close the file.
8. Open the file in in mode, call read() to read the record from the file and now call display() to display the record and at the end, close the file.
9. STOP

→ **Input-Output**

(i) **Input**

Enter the number of records = 3
Enter the records:-

- abc
- pqr
- xyz

(ii) **Output**

1. Name : abc
   Roll no: 01
2. Name : pqr
   Roll no: 02
3. Name : xyz
   Roll no : 03

→ Test Cases

S. no.	Step	Expected output	Actual Output	Result
1.	Accept data	Enter the data:	Program ended with exit code: 0	Fail
2.	Accept data	Enter the data:	Enter the data:	Pass
3.	Display Data	Name : abc Roll no : 01	--BLANK SCREEN--	Fail
4.	Display Data	Name: abc Roll no : 01	Name: abc Roll no: 01	Pass
5.	Exit	End of program	End of program	Pass

→ Conclusion

Thus, we successfully opened, closed files and read the data in them & also wrote in the files by just using C++ language.

5/12

CG Assignment - C7

→ **Title**

To write a C++ program for playing the game of Tic Tac Toe.

→ **Objectives**

To understand the basics of computer graphics & applying the concept of polymorphism, inheritance, various techniques for implementing line-drawing using object oriented programming

→ **Problem Statement**

Write a C++ program to implement the game of Tic Tac Toe. Apply the concept of Polymorphism.

→ **Outcomes**

i) Understand and apply the concepts of polymorphism for implementing reusable programming codes.

ii) Analyze and apply computer graphics for line-circle drawing, scan conversion and filling the polygon made with its help using object oriented programming concepts

→ **Software & Hardware Requirements**

i) **Software** : 64-bit open source Linux or its derivative, Open-source C++, Programming tools like g++/ gcc, OPENGL

ii) **Hardware** : Processor → intel(i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

Tic Tac Toe

Tic Tac Toe is a game where two players 'X' and 'O' fill the straight hash (#) shaped box ; which consists of two vertical lines crossing two horizontal lines, with their alternate turns.

- The player who fills the box with 3X's or 3O's in a horizontal, vertical or diagonal manner wins the game.
- In some cases, when none of the players succeed in filling the boxes horizontally, vertically or diagonally with 3X's or 3O's, then the game will be considered a draw.

O	X	
O	X	X
O		

'O' wins!

X		O
	X	O
	O	X

'X' wins!

X	O	O
O	X	X
X	O	O

Draw!

- For creating a Tic Tac Toe game in C++, we need to use arrays, functions & True-False condition
- For this, we need to create a two-dimensional array of 3X3 matrix of character type, we have to initialize it with elements from 1 to 9
- Once the player enters the digit, that number needs to be replaced by the symbol.
- After every turn, check the condition for winning or drawing of the game.

→ **Algorithm**

1. START
2. Declare a two-dimensional array to make the tic tac toe board.
3. Display the current status of the board after the turn of every player.
4. The first move is always made by 'X'. Display the options 1 to 9 on the screen for the player to select the position for entering the symbol.

1	2	3
4	5	6
7	8	9

5. After the number is selected, replace it by the specific symbol & display the board

X		

→

	2	3
4	5	6
7	8	9

6. If a horizontal, vertical or diagonal line is filled by the same symbol, that player wins.

X	X	X

	X	
	X	
	X	

X		
	X	
		X

If all the spaces have been filled, declare that the match is drawn.

STOP

→ Test Cases

Sr. no.	Step	Expected output	Actual output	Result
1.	X's turn	X | | | | | |	X | | | | | |	Pass
2.	O's turn	X | O | | | | |	X | O | | | | |	Pass
3.	X's turn	X | O | X | | | |	X | O | X | | | |	Pass
4.	O's turn	X | O | X | O | | |	X | O | X | O | | |	Pass
5.	X's turn	X | O | X | O | X | | X WINS	X | O | X | O | X | | X WINS	Pass

→ Conclusion

Thus, we have successfully implemented the Tic Tac Toe game using C++ programming language.

6/12

OOP
Assignment - B5

→ **Title**

To write a function template for selection sort that sorts various types of data.

→ **Objectives**

To understand and apply various methods & techniques for applying the sorting algorithms & templates using object oriented programming in C++.

→ **Problem Statement**

Write a function template for selection sort that inputs, sorts and outputs an integer array, and a float array.

→ **Outcomes**

i) Understand and apply the concepts like inheritance, polymorphism, exception handling, generic structures for implementing reusable programming codes.

ii) To understand various sorting algorithm and use of templates.

→ **Software & Hardware Requirements**

i) **Software** : 64-bit source linux or its derivatives, open source C++, G++/GCC tools, OPENGL

ii) **Hardware** : Processor → intel (i5)
storage → 1 TB
RAM → 4 GB

→ Theory

Selection sort

In selection sort, scan the array to find its smallest element and swap it with the first element.

- Then, starting with the second element, scan the entire list to find the smallest element and swap it with the second element.
- Then starting with the third element, the entire list is scanned in order to find the next smallest element. Continuing in this manner, we can sort the entire list.
- Generally, on pass i ($0 \leq i \leq n-2$), the smallest element is searched among last '$n-i$' elements and swapped with $A[i]$
- The list gets sorted after $n-1$ passes.

eg. Consider elements 70, 50 & 30

Initially,

50	70	30
A[0]	A[1]	A[2]
↑ Min	↑ i	

30	70	50
↑	↑ i, Min	↑ smallest element

30	50	70

Thus, the given list of elements is sorted successfully.

Templates

Generic programming is a technique that allows to write the code for any data type elements. The template is used as a tool for generic programming

eg.

```
int max(int left, int right)
{
if(left< right)
return right;
else
  return left;
}
```

```
double max(double left, double right)
{
if(left< right)
    return right;
else
    return left;
}
```

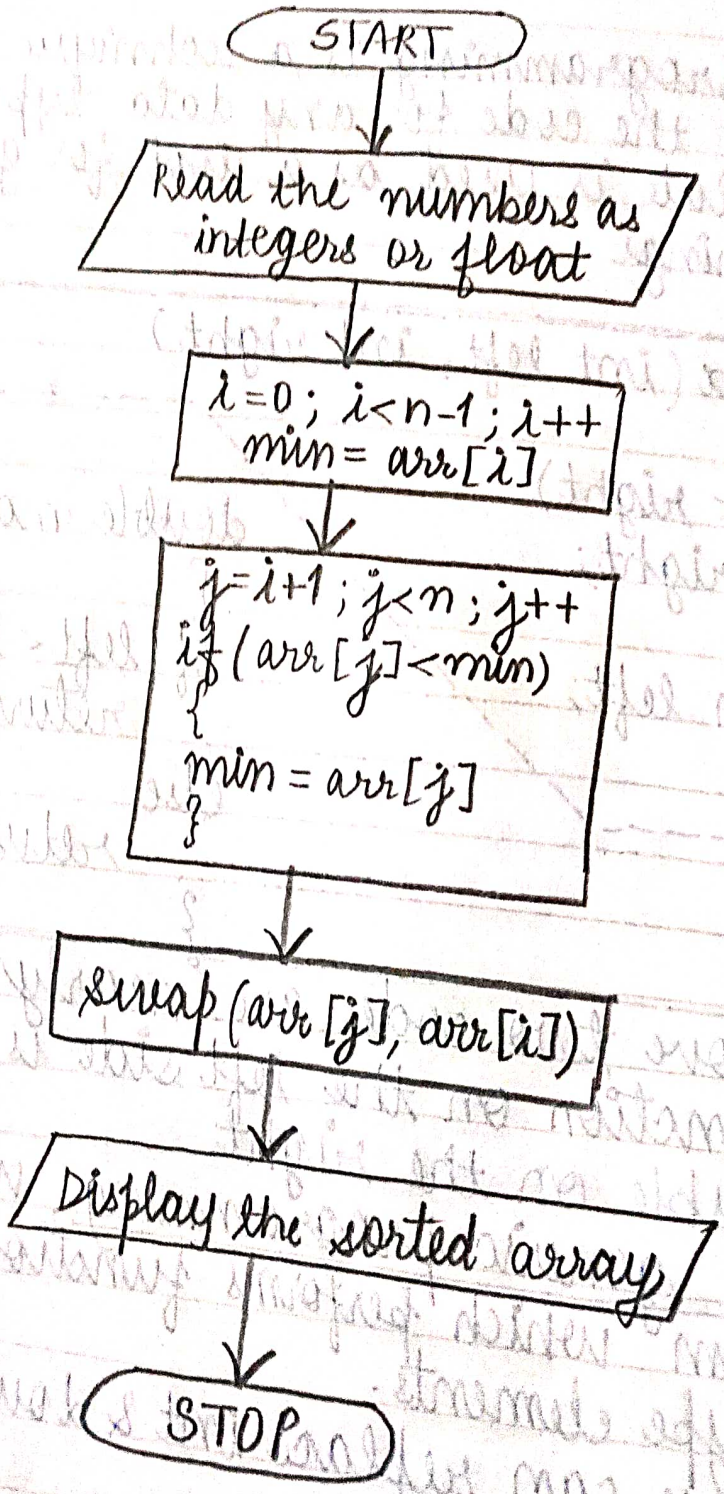
- The above two codes are nearly identical but the function on the left side is for integers & for double on the right.
- But in generic programming, we can write an algorithm which performs functionality of both data type elements.
- Hence, we can replace int & double by 'T.'

→ Syntax

```
template< typename T, typename...>
T max(...)
{
  :
```

- We can also write class instead of typename

→ Flowchart


START
Read the numbers as integers or float
i=0; i<n-1; i++
min= arr[i]
j=i+1; j<n; j++
if (arr[j]<min)
{
min = arr[j]
}
swap (arr[j], arr[i])
Display the sorted array
STOP

→ Algorithm

1. START
2. Read the numbers as integers or float values.
3. Sort according to ascending values.
   (i) Set min to location 0
   (ii) Search min element in the array.
   (iii) Swap the minimum value at location min
   (iv) Increment min to point to next element
   (v) Repeat until list is sorted.
4. Print sorted array as output
5. STOP

→ Input & Output

- Input

Enter the elements :-

54 78 14 71 1 9 37

- Descending Output - X

The unsorted list is

54 78 14 71 1 9 37

The sorted list is

78 71 54 37 14 9 1

- Ascending Output - ✓

The unsorted list is

54 78 14 71 1 9 37

The sorted list is

1 9 14 37 54 71 78

→ Test cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Sort	Unsorted list: 5 7 9 13 11 1 Sorted list: 1 5 7 9 11 13	Unsorted list: 5 7 9 13 11 1 Sorted list: 13 11 9 7 5 1	Fail
2.	Sort	Unsorted list: 5 7 9 13 11 1 Sorted list: 1 5 7 9 11 13	Same as the expected output	Pass
3.	Sort	Unsorted list: 2 9 7 5 6 Sorted list: 2 5 6 7 9	Same as the expected output	Pass

→ Conclusion

Thus, we have successfully implemented selection sort using templates in C++ programming language.

6/12

OOP Assignment - C6

→ **Title**

To write a program using STL for sorting & searching user defined records.

→ **Objectives**

To understand object oriented programming concepts like polymorphism, standard template library, etc.

→ **Problem statement**

Write a C++ program using STL for sorting and searching user defined records such as personal records (Name, DOB, Telephone Number, etc.) using vector container

→ **Outcomes**

(i) Understand & apply the concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable codes.

(ii) To learn about standard template library & use it to shorten long codes.

→ **Software & Hardware Requirements**

(i) **Software**: 64-bit open source Linux or its derivative, Open-source C++, Programming tools like G++/GCC, OPENGL

(ii) **Hardware**: Processor → intel (i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

STL

The standard template library (STL) is a collection of well structured generic C++ classes (templates) and functions. Basically, STL contains three basic components-

1. Container

The container is a collection of objects of different types. These objects store the data. The container can be implemented using template classes. There are two types of containers:-

(i) Sequence container

(ii) Associative container

2. Iterator

The iterators are basically objects but sometimes they can be pointers and hence iterators specify the positions in container. The iterators are used to traverse the contents of the containers.

3. Algorithm

The algorithms are used to process the contents of the container. The functionalities provided in the container are not sufficient to perform complex operations, hence the algorithms are used to support more complex operations for the containers.

→ Vectors

Vector is the most general purpose container. It stores the elements in contiguous memory locations. Any element can be accessed using index of it.

The index is given in subscript operator [ ]. It supports dynamic array. The memory allocation for vector is done at the run time.

→ Searching Algorithms

There are various functions available for searching an element from the list such as find search, binary search.

- The header file <algorithm> should be inserted in the C++ program to make use of algorithms as stated above.
- Various searching and sorting algorithms provided in the STL are:-

(i) sort()

Using quick sort, the elements are sorted

(ii) stable_sort()

Using stable sort method, the elements are sorted.

(iii) merge()

This function is used for merging the elements.

(iv) sort_heap()

Performs sorting on already created heap.

(v) min()

It finds the minimum element.

(vi) max()

It finds the maximum element.

(vii) binary_search()

It performs binary search on the sorted elements to search for a particular element.

→ Algorithm

1. START
2. Provide the user with seven options to choose from. and after selection of each operation, the menu should be provided again.
3. If selected option is 1, Create a new record, and ask the user to enter the total number of students.
4. If selected option is 2, Display the student record, displaying the roll number, name & dob for each student in a new line.
5. If selected option is 3, insert a new student into the record.
6. If selected option is 4, delete the student from the record by asking the roll number.
7. If selected option is 5, search the student from the list of given students.
8. If selected option is 6, sort the roll numbers of students in ascending order
9. If selection of option is 7, end the program.
10. STOP

→ Input

(i) Name : AA
Roll no : 1
DOB : 5/5/2003

(ii) Name : AB
Roll no : 2
DOB = 1/1/2003

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Create	Enter total no. of students:	Enter total no. of students:	Pass
2.	Insert	Enter roll no:	Enter roll no:	Pass
3.	Delete	Enter student roll no. to delete:	Enter student roll no. to delete:	Pass
4.	Search	Enter student roll no. to search:	Enter student roll no. to search:	Pass

→ Conclusion

Thus, we have successfully created a C++ program using STL for searching & sorting user defined records.

6/12

OOP Assignment - C7

→ **Title**

Write a C++ program to use map associative container.

→ **Objectives**

To understand the concepts of standard template library & use map associative container for various applications.

→ **Problem statement**

Write a program in C++ to use map associative container. The keys will be names of states & the values will be population of states. When the program runs, user is prompted to type the name of a state. The program then looks in the map, using the state name as an index & returns the population of the state.

→ **Outcomes**

To understand and apply concepts like inheritance, polymorphism, exception handling and generic structures for implementing reusable programming codes.

→ **Software & Hardware Requirements**

(i) Software : 64-bit open source linux or its derivative, open-source C++, programming tools like G++/GCC, OPENGL

(ii) Hardware : Processor → Intel (i5)
Storage → 1 TB
RAM → 4GB

→ Theory

Map

The map is an associative container in which the elements are stored in the form of key value and mapped value. For example (1, a), (2, b), etc.

- In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key. The types of key and mapped value may differ.
- The duplicate values are not allowed in map. Hence, it possesses one to one relationship
- The header file <map> is included in program for creating & using the map object.
- The key value & mapped value are grouped together in member type value_type which is a pair type as

```
typedef pair< const, Key, T> value_type;
```

- The syntax for creating map is -

```
map< key, value > map_name
```

- Various operations that can be performed on map are-

(i) insert(pair) : Element is inserted into the map.
(ii) erase (iterator i) : The element pointed by the iterator is deleted from the map
(iii) void swap(map) : Swaps contents of the map.
(iv) void clear() : Clear the contents
(v) size_type size() : Returns the size of the containers

→ Algorithm

1. START
2. Give a header file to map associative container
3. Insert state name so that we get values as population of that state.
4. Use population map insert()
5. Display the population of the state
6. STOP

→ Input and Output

- Input

Information, such as state name to map associative container.

eg. Maharashtra
Karnataka
Kerela

- Output

Size of the population of the given state chosen from the country

eg. (Maharashtra, 129652700)
(Karnataka, 85276760)
(Kerela, 36513432)

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Insert	Enter your choice:	Enter your choice:	Pass
2.	Delete	Enter key to be deleted:	Enter key to be deleted:	Pass
3.	Size	The size of the set is:	The size of the set is:	Pass
4.	Search	Enter key for searching:	Enter key for searching:	Pass
5.	Display	Elements of MAP are: [1,a], [2,b]	Elements of MAP are: [1,a],[2,b]	Pass

→ Conclusion

Thus, we have successfully implemented map associative container using C++

6/12

CG
Assignment - B5

→ **Title**

To apply logic and generate various fractal patterns using Koch curves.

→ **Objectives**

To understand basics of computer graphics, apply various methods & techniques for implementing line-circle drawing, projections, animation, shading, illumination & lighting.

→ **Problem statement**

Write a C++ program to generate snowflakes using concept of fractals, to generate fractal patterns using Koch curves.

→ **Outcomes**

i) Analyze and apply computer graphics algorithms for line-circle drawing, scan conversion and filling it up with the help of object oriented programming concepts

ii) Apply logic to implement curves, fractals, animation and gaming programs.

→ **Software & Hardware Requirements**

i) Software : 64-bit open source ~~Linux~~ or its derivatives, open source C++, Programming tools like G++/GCC, OPENGL

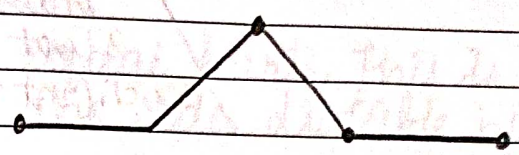
ii) Hardware : Processor → Intel (i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

Koch snow Flake/Triadic Curve

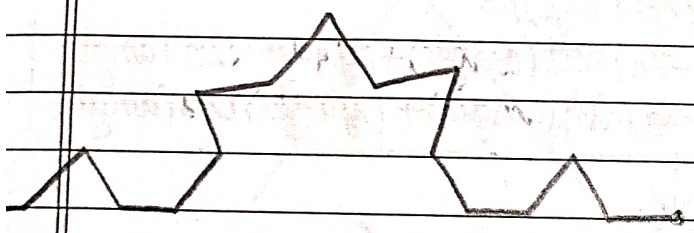
The koch snowflake is also called koch curve.

- The Koch curve can be drawn by dividing a line into 4 equal segments with scaling factor 1/3 and middle two segments are so adjusted that they form adjacent sides of an equilateral triangle as shown.



1st Approximation

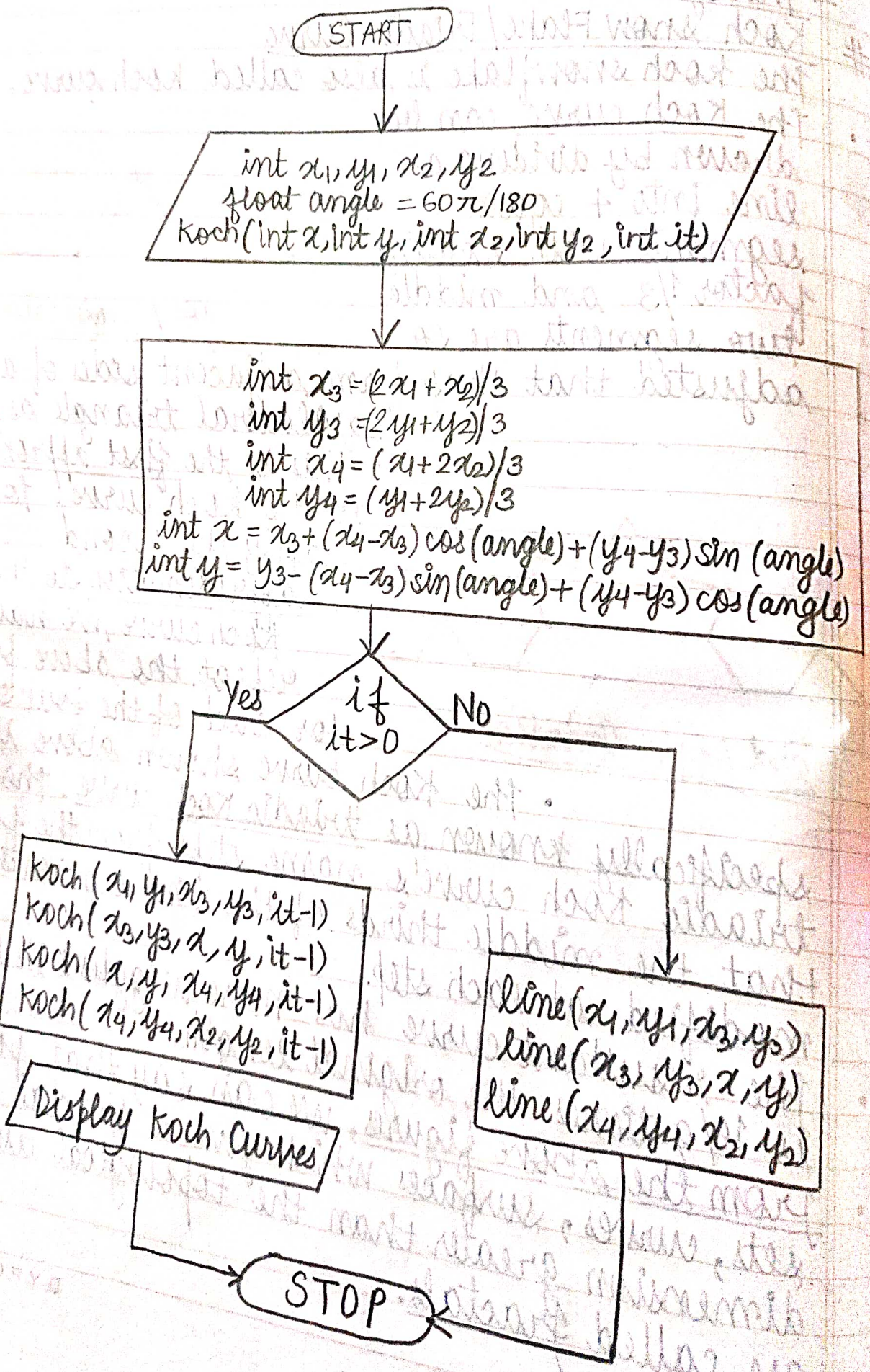
- This is the first approximation to the koch curve. To apply the second approximation to the koch curve, we have to repeat the above process for each of the four segments.



2nd Approximation

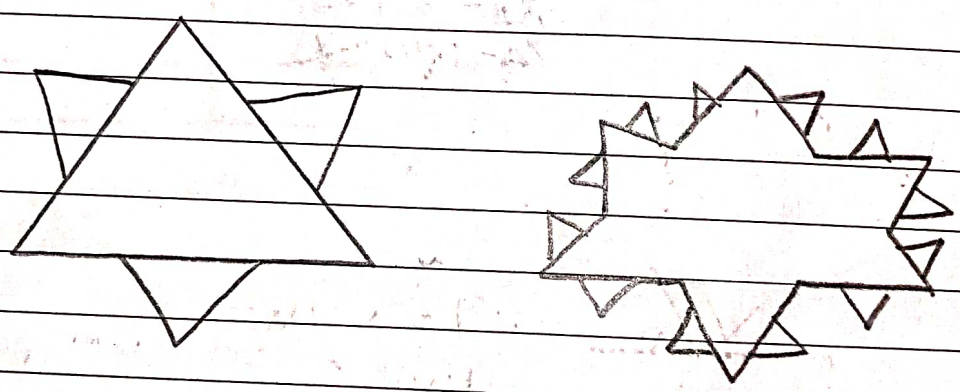
- The Koch curve shown above is more specifically known as triadic Koch curve. The triadic Koch curve's name steps from the fact that the middle thirds of the line segments are modified at each step.
- The resultant curve has more wiggles & its length is 16/9 times the original length.
- From the above figures, we can say that point sets, curves, surfaces which give a fractal dimension greater than the topological dimensions are called Fractals.

→ Flowchart


START
int x1, y1, x2, y2
float angle = 60π/180
koch(int x, int y, int x2, int y2, int it)
int x3 = (2x1 + x2)/3
int y3 = (2y1 + y2)/3
int x4 = (x1 + 2x2)/3
int y4 = (y1 + 2y2)/3
int x = x3 + (x4 - x3) cos(angle) + (y4 - y3) sin(angle)
int y = y3 - (x4 - x3) sin(angle) + (y4 - y3) cos(angle)
if it > 0
Yes
No
koch(x1, y1, x3, y3, it-1)
koch(x3, y3, x, y, it-1)
koch(x, y, x4, y4, it-1)
koch(x4, y4, x2, y2, it-1)
line(x1, y1, x3, y3)
line(x3, y3, x, y)
line(x4, y4, x2, y2)
Display Koch Curves
STOP

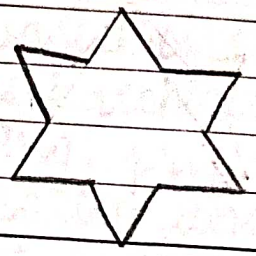
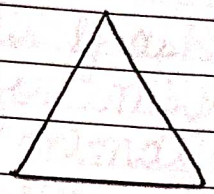
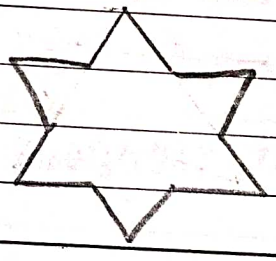
→ Algorithm

1. START
2. Draw an equilateral triangle. It's best if the length of the sides are divisible by 3 because of the nature of the fractal
3. Divide each side in three equal parts. This is why it is handy to have the sides divisible by three.
4. Draw an equilateral triangle on each middle part. Measure the length of the middle third to know the length of the sides of these new triangles



5. Divide the outer side into thirds. You can see the 2nd generation of triangles covers a bit of the first. These three line segments shouldn't be parted in three.
6. Draw an equilateral triangle on each middle part in the third approximation until you are satiated with the design of the snowflake
7. STOP

→ Test Cases

S. no.	Step	Expected Output	Actual Output	Result
1.	Koch curve			Fail
2.	Koch curve		Same as the expected output	Pass

→ Conclusion

Thus, we have successfully drawn Koch's curve or snowflake using graphics

6/12

CG
Assignment - C6

→ **Title**
To make use of OPENGL in order to draw patterns of sunset & sunrise.

→ **Objectives**
To understand basics of computer graphics, apply various methods & techniques for implementing line-circle drawing, projections, animation, shading, illumination and lighting using the concepts of Object Oriented Programming

→ **Problem statement**
Write OPENGL program to draw sunrise and sunset

→ **Outcomes**
i) Analyze & apply computer graphics algorithms for line-circle drawing, scan conversion & filling with the help of object oriented programming concepts.
ii) Apply logic to implement, curves, fractals, animation and gaming programs.

→ **Software & Hardware Requirements**
(i) Software : 64-bit open source linux or its derivative, Open source C++, Programming tools like G++/GCC, OPENGL
(ii) Hardware : Processor → intel(i5)
Storage → 1 TB
RAM → 4 GB

→ Theory

OPENGL

Open Graphics Library (OPENGL) is a cross-language (language independent), cross platform (platform independent) API for rendering 2D and 3D vector graphics (use of polygons to represent image). OPENGL API is designed mostly in hardware.

→ Design

This API (Application Programming Interface) is defined as a set of functions which may be called, by the client program.

→ Development

It is an evolving API and the Khronos group regularly releases its new version having some extended feature compared to previous one.

→ Associated Libraries

The earliest version is released with a companion library called OPENGL utility library. But since OpenGL is quite a complex process, so in order to make it easier other libraries such as OpenGL Utility Toolkit is added which is later superseded by free glut. Later included libraries were GLEE, GLEW and gliding.

→ Implementation

Mesa 3D is an open source implementation of OpenGL. It can do pure software rendering and it may also use hardware accelaration on BSD, Linux, and other platforms by taking advantage of Direct Rendering Infrastructure

→ Header Files

For graphics programs, we include the header file for the OpenGL core library.

- For most applications, we will also need GLU. For instance, with Microsoft Windows, we need to include the header file (windows.h) for the window system.
- This header file must be listed before the OpenGL and GLU header files because it contains macros needed by Microsoft Windows version of OpenGL libraries. So the source file in this case would begin with-

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
```

- Thus, we use GLUT to handle the window-managing operations, we do not need to include gl.h & glu.h because GLUT ensures that these will be included correctly. Thus, we can replace the header files for OpenGL and GLU with

```
#include<GL/glut.h>
```

- We could include gl.h & glu.h as well, but doing so would be redundant and could affect the program portability. In addition, we will need to include header files that are required by the C++ code.
- For example-

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h> & of course
#include<iostream.h>
```

→ Algorithm

1. START
2. Do the declarations

```
float ballX = -0.8f;
float ballY = -0.3f;
float ballZ = -1.2f;
float colR = 3.0;
float bgColR = 0.0;
```

and so on...

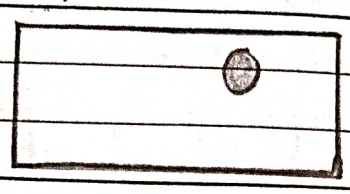
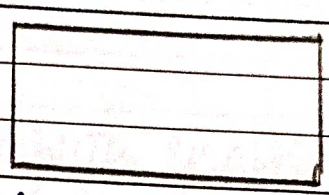
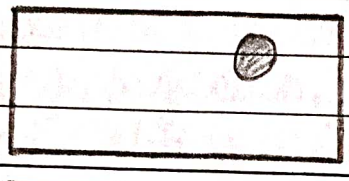
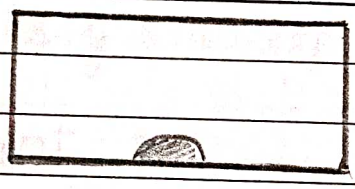
3. The sun would be drawn perfectly round in shape. set the colour of the sun.
4. Begin to draw the sun point by point until the whole circle is drawn.
5. Start moving the sun from one end of the screen from behind the mountains and the clouds and to the other side of the console to give the illusion of a sunset.
6. Start rendering

```
glEnable (GL_DEPTH_TEST);
glEnable (GL_COLOR_MATERIAL);
glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);
glEnable (GL_LIGHT1);
glEnable (GL_NORMALIZE);
```

7. Run the program. Make sure the following header files are included in the program-

(i) #include <openGL/openGL.h>
(ii) #include <GLUT/GLut.h>

→ Test Cases

S. no	Step	Expected Output	Actual Output	Result
1.	sun pattern			Fail
2.	sun pattern		Same as the expected output	Pass
3.	Sun pattern		Same as the expected output	Pass

→ Conclusion

Thus, we have successfully created sunrise and sunset using OPENGL.

6/12

Mini Project

→ **Title**

To make a working analog clock using graphic library functions available in C++ programming language

→ **Objectives**

To understand the basics of computer graphics, apply various methods & techniques for implementing line-circle drawing, projection, animation, shading, illumination, and lighting using concepts of Object Oriented Programming

→ **Problem statement**

Design & implement game/animation clip/ Graphics editor using open source graphics library. Make use of maximum features of Object oriented programming

→ **Outcomes**

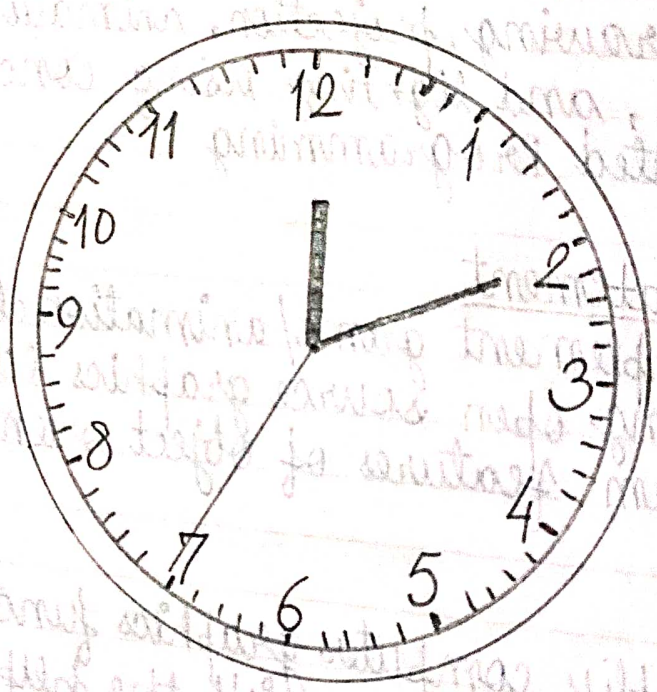
(i) Analyze & apply computer graphics functions for line-circle drawing, filling with the help of object oriented programming in C++

(ii) Apply logic to implement curves, fractals, animation and gaming programs.

→ **Software & Hardware Requirements**

(i) Software : 64-bit open source Linux or its derivative, Open source C++, Programming tools like G++/GCC, OPENGL

(ii) Hardware : Processor → intel (i5)
Storage → 1 TB
RAM → 4 GB



Analog Clock

→ Theory

Analog Clock

An analog clock shows time passing by moving hands continuously. The history & origin of the clock is an interesting one as it is hard to point which civilization invented it first.

People would originally determine the time of the day by observing the length of their shadows. When the shadow was the shortest, it was noon. Over the centuries, there have been sand dials, water clocks and other such clocks to tell the time. Once you can read the time, you can read a random analog clock and not cross-check to see a digital clock.

→ What is an Analog Clock?

An analog clock is a tool for reading the time of the day. The longest hand indicates the seconds the longer one indicates the minutes while the smallest one indicates the hours.

- A clock or watch that contains moving hands & hours with markings from 1 to 12 to show time is called analog clock.
- A day has 24 hours. A clock, having markings from 1 to 12, identifies all 24 hours. Some of the clocks have roman numerals (I, II, III, etc.) or some don't have any numbers at all! Random analog clocks use angles to tell time. They have hands rotating around the clock's face
- A.M. – ante meridiem
  P.M. – post meridiem

All C++ functions used

(i) Structures

A structure is a user defined data type in C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

- Syntax

```
struct structure_name {
    member 1;
    member 2;
    :
    member n;
};
```

(ii) Functions

A function is a set of statements that take input, do some specific computation, and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

- Syntax

```
return-type function_name(parameter(s))
```

eg. int calculatehrs(int h)

(iii) switch statement

switch case statement evaluates a given expression & based on evaluated value; it executes the statements associated with it. Basically, it is used to perform various tasks based on different cases.

- They are a substitute for long if statements that compare a variable to several integral values.
- Syntax

```
switch (expression)
{
case 1:  statement_1;
         break;
case 2:  statement_2;
         break;
   :
case n:
default: default statement;
}
```

(iv) if-else if

Here, a user can decide from multiple options. The C++ if statements are executed from the top down.

- As soon as one of the conditions controlling the if is true, the statement associated with that if is executed & the rest of the C++ else-if ladder is bypassed.
- If none of the conditions is true, then the final else statement will be executed.
- Syntax

```
if (condition 1)
{
}
else if (condition 2)
{
}
else
{
```

(v) return statement

The return statement returns the flow of the execution to the function from where it is called.

- Syntax

  return(expression);

- As soon as the statement is executed, the flow of the program stops immediately and returns the control from where it is called.

(vi) initgraph

initgraph() initialises the graphics system by loading a graphics driver from disk (or validating a registered driver), then putting the system in graphics mode.

eg. initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

(vii) setcolor

It is a function used to set the current drawing color to a new color.

- Syntax

  setcolor(COLOR_NAME);

(viii) gettime

gettime function gets the time in hours, minutes and seconds from the system.

eg. gettime(&t)

(ix) getbkcolor

This function returns the current background color.

- Syntax

  getbkcolor();

(x) gdriver

It is a pointer to an integer specifying the graphics driver to be used. In our programs, we use 'DETECT' for auto detection of the graphics driver.

eg. int gdriver = DETECT, gmode

(xi) gmode

Graphicsmode is a pointer to an integer that specifies the graphics mode to be used.

- If gdriver is set to DETECT, then initgraph sets gmode to the highest resolution available for the detected driver

(xii) pieslice

pieslice() draws & fills a pie slice with (x, y) centre & 'r' radius. The slice travels from s-angle to e-angle.

- Syntax

pieslice(int x, int y, int s-angle, int e-angle, int r);

(xiii) settextstyle

It is used to change the way text appears.

eg. settextstyle(int font, int direction, int font-size);

(iv) outtextxy

The outtextxy() function which displays the text or string at a specified point (x, y) on the screen.

- Syntax

outtextxy(int x, int y, char* string);

(xv) circle

circle() draws a circle with centre (x,y) & radius 'r'.

- Syntax

circle(x, y, radius);

(xvi) sleep

sleep() function allows the user to wait for a current thread for a specific time. Only the present executable will sleep, rest CPU functions will carry on normally.

eg. sleep(10);

// Program will sleep for ten seconds

(xvii) clearviewport

clearviewport() function clears the current view port and resets the current position to (0,0), relative to the viewport.

eg. clearviewport();

(xviii) exit

exit() terminates the calling process without executing the rest of the code.

eg. exit(0);

(xix) void

void functions are known as non-returning functions. They are 'void' due to the fact that they are not supposed to return values. Although, a void function can still return even if its not allowed to have a return type.

→ Algorithm

(i) START

(ii) We know that a circle has 360 degrees & since a clock is circular in shape, it will have 360 degrees as well. So, since there are 12 hours on a clock, each hour will be 30 degrees apart from each other.

(iii) Similarly, as there are 60 minutes in an our, each minute will be 360/60 = 6 degrees apart. Make a switch case for all the cases.

(iv) Initialize graphics by using initgraph, gdriver and gmode.

(v) Use the gettime() function to fetch the current time of the system. Since the reading of time & execution of program has a delay of nearly 2 seconds, you can add 2 seconds into the seconds counter to make it exact.

(vi) Now, start by assigning the background colour to black by CODE '0'. Then, settextstyle & setcolor for succeeding outtextxy to print the heading 'Analog Clock'

(vii) Now, use circle function to draw shape of the clock & to mark 1, 2, 4, 5, 7, 8, 10, 11 hours & write 3, 6, 9, 12 by text at appropriate position.

(viii) Draw hour, minute & second hands with distinguishable colours & sizes (lengths).

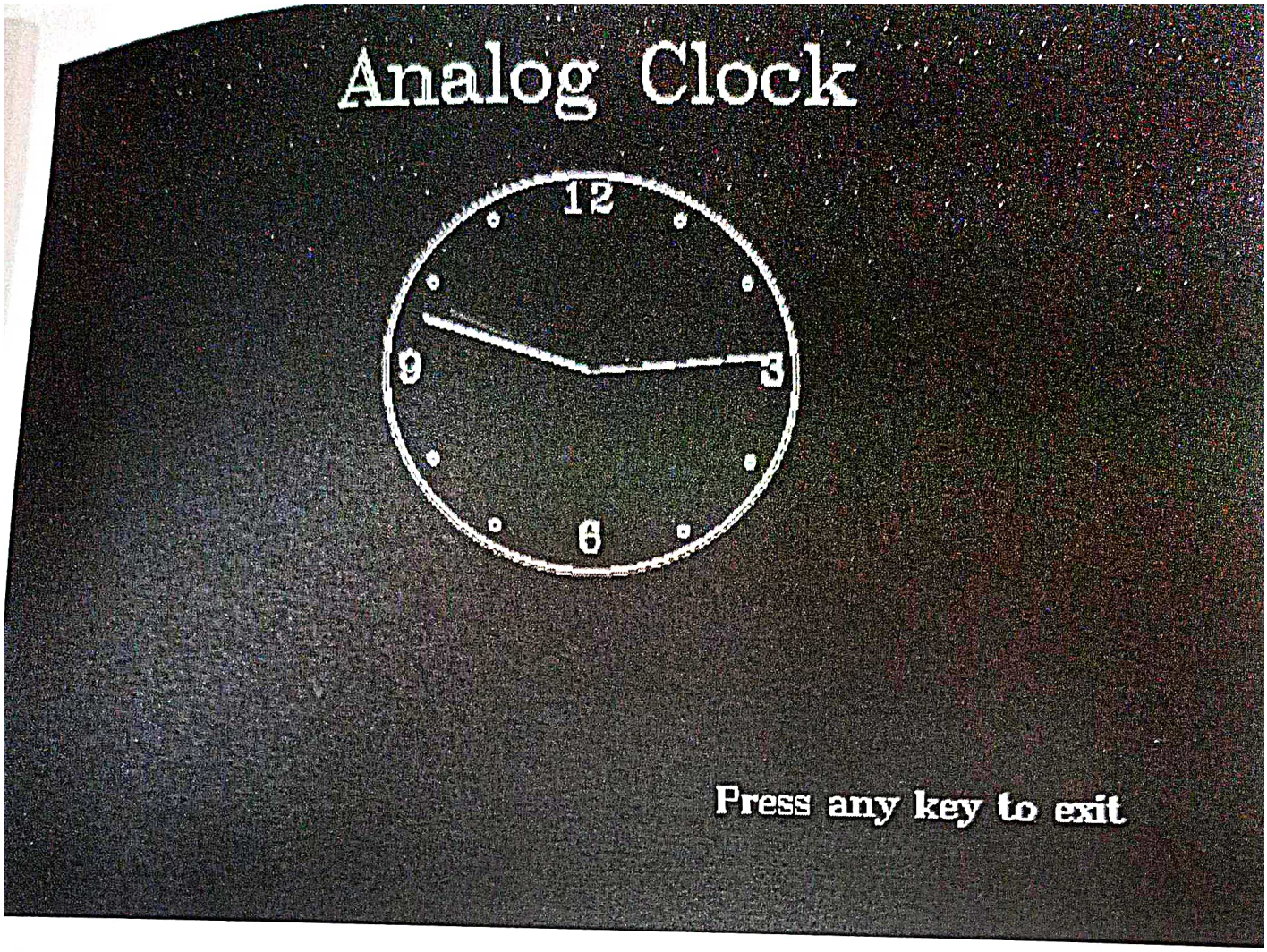
(ix) Display "Press any key to exit" on the bottom right corner of the screen & if any key is pressed, display "Thank you" before waiting for sleep() time

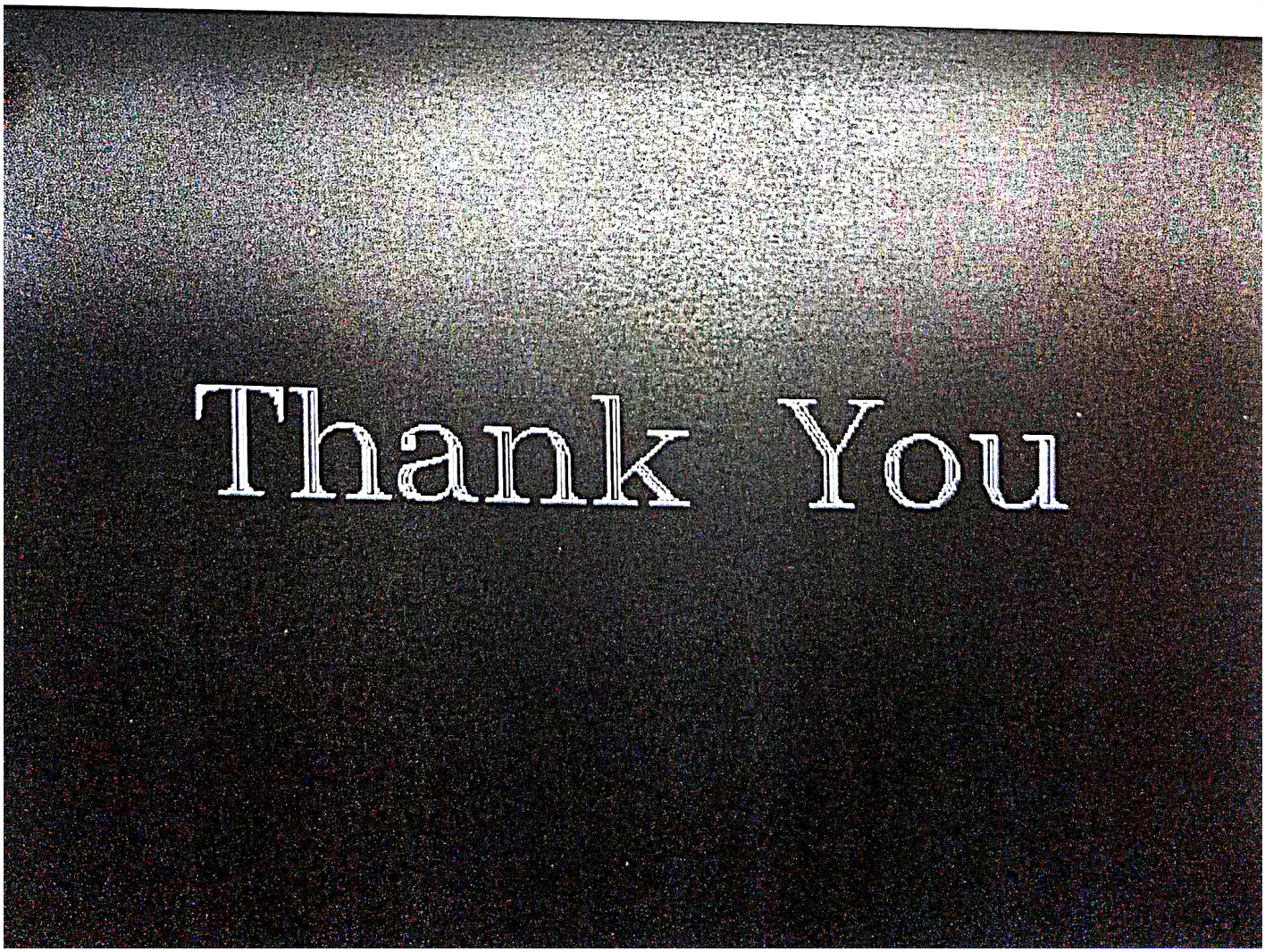
(x) STOP

→ Conclusion

Thus, we have successfully created a working analog clock which matches with the current time to seconds. using the graphics library functions available in C++.

Rita
15/12


Analog Clock
12
3
6
9
Press any key to exit


Thank You