SL NO.	Date	Name of Experiment	Page No.	Remarks
1.		Function point value	1-6	
2.		Constructive cost model (COCOMO	7-9	
3.		Cyclomatic - complexity	10-13	
4.		Software Maturity Indext (SMI)	14-17	
5.		Di		

INDEX

Name of the Experiment: 1. Function Point Value. | Page no. 1

Program Statement:-

Design and develop a java program to calculate Function point value.

Program-Description:-

* Function point is an element of software development which helps to approximate the cost of development early in the process. It may measures functionality from user's point of view.

* Scale varies from 0 to 5 according to character of complexity Adjustment factor.

F = 14 * scale

* calculate Unadjusted complexity adjustment factor

CAF = 0.65 + (0.01 * F)

* calculate Function point

$$FP = UFP * CAF$$

Algorithm:-

Step 1: Start the program

Step-2: create class called FP value.

Step-3: Declare and initialize neccessary variables and arrays within the FP value class.

Step-4: create a constructor method with in FP-class to initalize the variables.

Step-5: create an input method within the FPclass to accept user input for the counts and average complexity adjustments

Step-6: createa process() method within the FPvalue class to calculate the function point value.

Step-7: create FPVsiree class, invoke input(), process and output(), this is the main method.

Output:-

```
Enter number of external inputs:
2
Enter number of external outputs:
3
Enter number of external inquires:
1
Enter number of internal logical files:
4.
Enter number of external interfaces:
1
2
1
3
2
Enter sum of value adjustments factors(fi):
2
Count total is: 24
Sum of fi is : 2
FP = count total * (0.65 + (0.01 * sum of fi))
Functional point (FP) value is : 16.56.
```

Conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to calculate the estimated effort using cocomo model.

Description:-

*The COCOMO (Constructive cost model) is well known software cost estimation model. It is based on the principle that the effort required to develop a software project is proportional to its size and complexity.

Algorithm:-

step-1: start the program.

step-2: create a class called cocomo.

step-3: declare neccessary variables for screens, reports, components, complexity weights and productivity factor.

step-4: read the inputs from the user and store into the variables.

step-5: calculate the Estimated Effort by the dividing the NOP by the productivity factor.

step-6: display the EE and NOP values.

output:-

```
Enter no of screens:
10
Enter complexity weights:
4
Enter no of Reports:
5
Enter complexity weight:
3
Enter no. of components:
8
Enter complexity weight
5
Enter value of prod:
150
NOP=104
```

The Estimatted effort for simple ATM using cocomo II model is: 0.0933334.

conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to calculate the cyclomatic complexity of a control graph.

Description:-

* Cyclomatic complexity is a way to measure the complexity of software modules. It is based on the control flow graph of the program, which represents the flow of execution between various statements, branches, and loops.

$$M = E - N + 2$$

→ 'M' represents the cyclomatic complexity.

→ 'E' is the number of edges in the control flow graph.

→ N is the number of nodes in the control flow graph.

Algorithm:-

step-1: Start the program.

step-2: Create a class called cyclomatic.

step-3: Declare variables for the number of edges(e), number of nodes(n), and cyclomatic complexity(V).

step-4: Read the number of edges(e) from the user.

step-5: Read the number of nodes(n) from the user.

step-6: calculate the cyclomatic complexity(V) using the formula $V=e-n+2$

step-7: Display the calculated cyclomatic complexity(V).

step-8: End the program.

Output:-

```
Enter the number of edges:
12
Enter the number of nodes:
10
The cyclomatic complexity of a graph is:
4
```

conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to calculate SMI value.

Description:-

* The Software Maturity Index(SMI) is a metric used to access the maturity level of a software product or system. It provides a quantitative measure of the software development process and the quality of the resulting software.

* the SMI value indicates the degree of stability, reliability, and predictability of the software product.

SMI = MT - (FA+FC+FD) / MT ;

SMI - software Maturity index value

MT - no. of software functions /modules in current

FC - no. of software functions / modules that contain changes from the previous release.

FD- no. of functions/modules that are deleted from the previous release.

FA- no. of functions/modules that are added from the previous release.

Algorithm:-

step-1: start the program.

step-2: create class called SMI

step-3: Declare variables for Mean Time (MT), Failures avoided (FA), Failures corrected (FC), Failures deleted (FD), and software maturity Index(SMI)

step-4: Read the value of MT, FA, FC, FD from the user.

step-5: calculate SMI using the formula

SMI = (MT-(FA+FC+FD))/MT

step-6: display the calculated SMI.

step-7: End the program.

Output:-

```
Enter MT:
100
Enter FA:
20
Enter FC:
10
Enter FD:
5
The resultant SMI is: 0.65
```

conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to draw line by using DDA.

Description:-

* The Digital Differential Analyzer (DDA) is a line drawing algorithm.
* It is a simple and efficient algorithm that calculates the co-ordinates of pixels along a line segment based on the slope of the line.
* The DDA Algorithm works by incrementally calculating the x and y co-ordinates of pixel on the line.

Algorithm:-

step-1: start the program.

step-2: create class called 'dda' that extends the 'Applet' class and implements the 'Actionlistener' interface.

step-3: Declaring variable for the starting and ending co-ordinates of the line 'x_1', 'x_2', 'y_1', 'y_2'.

step-4: create lables (L_1, L_2, L_3, L_4) and text field (T_1, T_2, T_3, T_4) and button (b)

step-5: implement init() method to initialize the applet by adding labels.

step-6: implement 'paint()' to perform the line drawing.

step-7: paint() method can retrive the co-ordinate values x, y.

step-8: implement 'Actionformed()' to perperom or handle the button clicked event. To call Repaint() method calls paint()

step-9: End the program.

output :-

$x_1 - 200$ $x_2 - 300$ $y_1 - 100$ $y_2 - 150$ draw

conclusion :-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to draw a line by using Bresanham's Line Drawing Algorithm.

Description:-

* Bresanham's line Algorithm is an efficient method for rasterizing lines on a grid based display, such as a computer screen. It determines which pixels to activate to represent a line between two given points.

* Bresanham's algorithm uses the decision parameter when the increment or decrement the y-co-ordinate and adjust the error term accordingly. By making efficient choices based on the error term.

Algorithm:-

Step-1: Start the program.

Step-2: create a class called 'bresan' that extends the 'Applet' class and implements the 'ActionListener' interface.

Step-3: Declare variables L1, L2, L3, L4 as Label and textfields T1, T2, T3, T4 as TextFields. and button 'b'.

Step-4: implement init() method to setup the user interface by adding labels, textfields, button.

Step-5: parse the user input co-ordinates from the textfield to integers

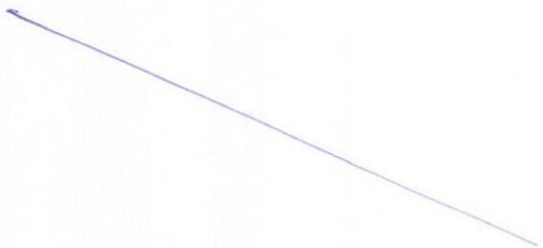
Step-6: calculate the difference between x and y co-ordinates.

Step-7: implement Bresanham's line algorithm using a loop to calculate and plot the co-ordinate of each pixel along the line.

Step-8: End the program.

Output:-

$x_1 - 20$ $x_2 = 230$ $y_1 = 320$ $y_2 = 140$ Draw



conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to draw a mid point circle.

Description:-

* The mid point circle algorithm is an efficient method for drawing circles on raster display. only using integer arthematic.

* this algorithm works by considering a circles symmetry and utilizies the concept of the midpoint to determing which pixels to plot to approximate circle. It avoids the need for expensive trignometric calculations or floating point, making it computationally efficient.

Algorithm:-

Step-1: Start the program.

Step-2: Import the required AWT classes and packages.

Step-3: Define a class named "circle" that extends the applet.

Step-4: Declare the neccessary variables and components such as labels, textfields, and a button.

Step-5: Override the paint() method to perform the circle drawing

Step-6: Retrives the values entered in the text fields for the center coordinates (xc, yc) and the radius (rad).

Step-7: Initialize variables x and y to 0 and rad respectively.

Step-8: Repeat the loop until x become greater than or equal to y.

Step-9: End the program.

Output:-

$x_0 = 100$ $y_0 = 150$ radius = 80 [draw]

conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program Mid point Ellipse.

Description:-

* mid point ellipse algorithm is a method used to draw an ellipse using raster graphics.

* Define a required variables to store the center co-ordinates (xc, yc) of the Ellipse, the major-axis length (a) and minor axis length (b)

* calculate the squre of a and b a-squre = a*a and b-sequre = b*b

*calculate the initial decesion parameter

P1 = b-Squre - a_squre * b + (0.25 * a-squre).

dx = 2 * b-Squre * x

dy = 2 * a-squre * y.

Algorithm:-

Step-1: Start the program.

Step-2: Initialize the required variables x, y, xc, yc, rx, ry, P_1 and P_2.

Step-3:- create labels and text fields for user input

Step-4: create a button for drawing the ellipse.

Step-5: Parse the user input for the semi-major axis (rx), semi-minor-axis (ry) and center co-ordinate (xc, yc).

Step-6: set the initial values of x and y to 0 and ry.

Step-7: calculate the initial value of P_1 using the equation $P_1 = (ry)^2 - rx^2 + rx^2/4$

Step-8: End the program.

output:-

Semi-major-axis-200 semi-minor-axis-300

x-130 y-160 draw.

(out)

conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to implement 2D-basic Transformation.

Description:-

* 2D-transformations refers to the manuplation of objects in a two-dimensional co-ordinate.
* These transformations are used to modify the position, size, orientation, and shape of 2D objects.
* These are several types of 2D Transformations.

1. Translation
2. Scaling
3. Rotation
4. Reflection
5. shearing

Algorithm:-

step-1 : start the program

step-2: create a java applet class named "twoD" that extends the applet class and implements the ActionListener interface.

step-3: Declare and initialize the neccary variables and components, including, labels, textfield and buttons.

step-4: implement the "init()" method to initalize the applet by adding labels, textfields and buttons to the applet GUI.

step-5: implement "paint()' method to handle drawing of the original line.

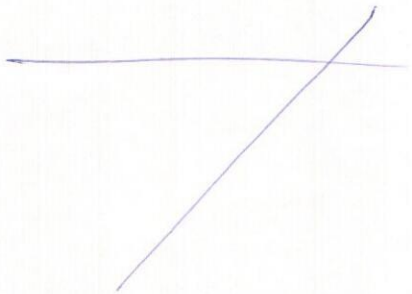
step-6: if button b1 clicked method, check the source of the event
button b2 perform the translation

step-7: End the program.

output:-

$X_1 - 100$ $X_2 - 10$ $Y_1 - 20$, $Y_2 - 200$ $X_3 - 100$, $Y_3 - 200$

Show Show



conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to implement Line clipping.

Description:-

* Line clipping is a technique used to determine which portions of line segment are visible and should be drawn on the screen.

* It is particularly useful when rendering lines that extends beyond the boundaries of a viewport or window.

* Determine the intersection points of the Line segment with the viewport boundaries by applying clipping tests.

Algorithm:-

Step-1: Start the program.

Step-2: Initialize the applet by adding a button and the neccessary event listeners.

Step-3: In the 'mousepressed' method, capture the starting coordinates of the line segment.

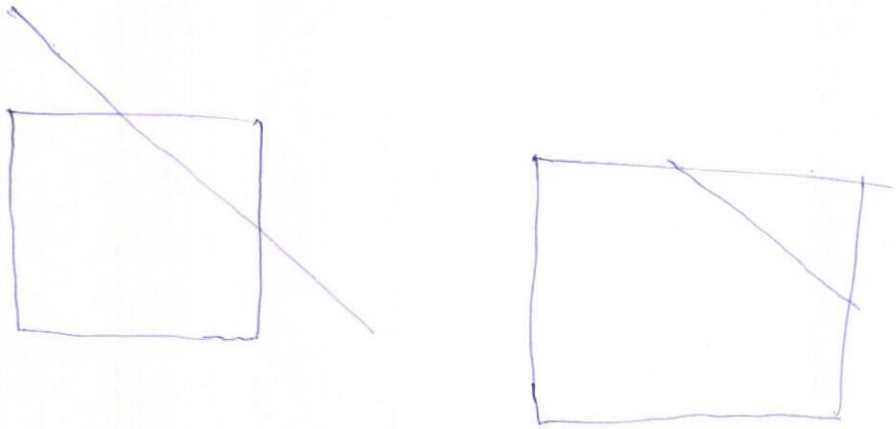
Step-4: In the 'mouseReleased' method, capture the ending co-ordinates of the line segment.

Step-5: In the paint() method, draw the rectangle region and the original line segment.

Step-6: In the 'actionperformed' method handle when the button clicked event.

Step-7: End the program.

output:-



Rectangular region.

conclusion:-

The above java program executed successfully.

Problem-defination:-

Design and develop a java program to implement cohen sutherland Line clipping Algorithm.

Description:-

* The most commonly used line clipping Algorithm is the cohen sutherland (Line clipping) Algorithm.

* which divides the 2D space into several regions based on the boundaries of the view port.

* Each region is represented by a 4-bit code that indicates whether a point is inside or outside of the viewport.

Algorithm :-

Step-1: Start the program

Step-2: Initialise the applet by adding a button and registering the neccessary event listeners.

Step-3: Implement the mouse event listener methods to handle mouse actions.

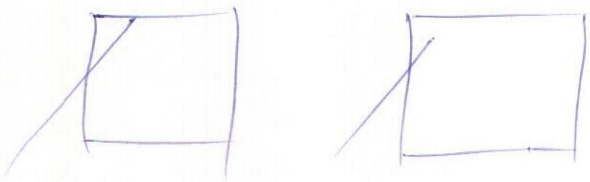
Step-4: In the mouse pressed method, capture the starting co-ordinates of the line segment.

Step-5: In the paint method, draw the rectangular region and the original line segment

Step-6: Implement the 'plot_clip' method to draw a dot at each visible co-ordinate.

Step-7: Stop the program.

output :-



-1.8

[0,0,0,0]

second [1,0,0,0]

second [1,0,0,0]

[0,1,0,1]

conclusion :-

The above java program executed successfully.

Name of the Experiment: 19. Clock | Page no. 60

Problem-Defination:-

Design and develop a java program to draw a clock.

Algorithm:-

Step-1: Start the program.

Step-2: Create a new java class that extends the 'Applet' class.

Step-3: Import the neccessary libraries for graphics and time handling such as java.util.*, java.awt.*

Step-4: Override the init() method of applet class. This method is called when the applet is initialized.

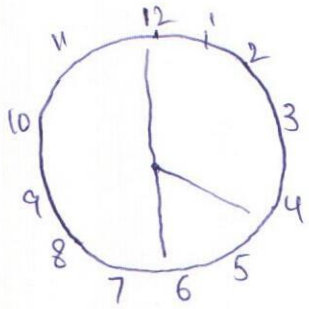
Step-5: Override the paint method of Applet class. called whenever the applet needs to be redrawn.

Step-6: Draw a clock face using drawOval(), draw the hour hand, minute drawLine(),

Step-7: End the program.

Output:-

hour [12] minute [30] second [20] [Start]



conclusion:-

The above java program executed successfully.

Problem-Defination:-

Design and develop a java program to draw wheel rotation.

Algorithm:-

Step-1: Start the program.

Step-2: Create class named wheelrot extends Applet class.

Step-3: Declare instance variables to store the wheel image, rotation angle, and other neccessary parameters

step-4: Override the init() method of the 'Applet' class. This method called when the applet is initialised.

step-5: override the paint() method of the Applet class. This method is called whenever the applet needs to be redrawn.

step-6: override the mouse dragged() method MouseMotionListener interface.

step-7: call the repaint() method to the trigger to redraw the applet.

step-8: End the program.

Problem-Defination:-

Design and develop a java program to implement Bezier curve.

Description:-

* Bezier curve is a type of curve commonly used for in computer graphics and geometric modeling.

* It is set of control points that influence its shape.

* The curve starts at the first control point and ends at the last control point.

* while the intermediate curve points determine the curve path.

* To draw bezier curve, various algorithms can be used.

Algorithm:-

Step-1: start the program

Step-2: Input the co-ordinates of the three control points $P_0, P_1, P_2 \ldots$

step-3: set the parameter t to a value between 0 and 1 (inclusive) to define the position alng the curve.

step-4: Compute the blending functions

$B_0(t) = (1-t)^2$

$B_1(t) = 2 * t * (1-t)$

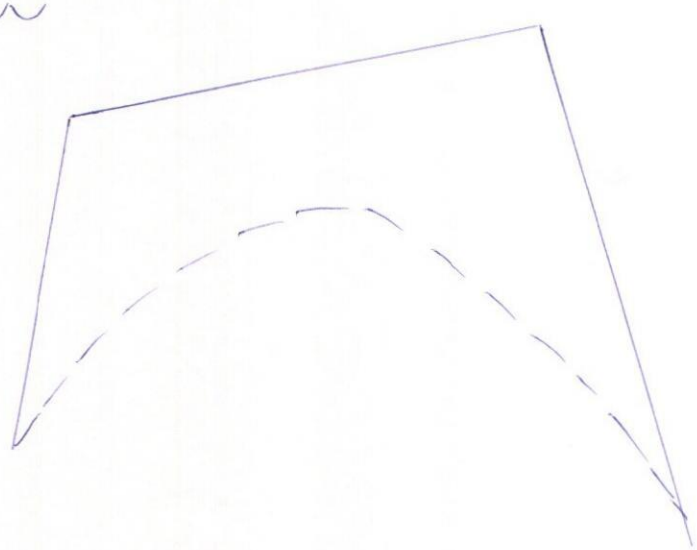
$B_2(t) = t^2$

Step-5: Repeat the steps 2-4 for different values of t to generate points along the curve.

Step-6: Connect the generated points to form the bezier curve

Step-7: End the program.

Output:-



conclusion:-

The above java program executed successfully.