

**[2D CYCLE]**

**S. Y. B. Tech Computer Engineering**

**Project 1 [BTECCE22402: Computer Graphics and Gaming]**

**By**

**Atharva Shevate**

**202201727**

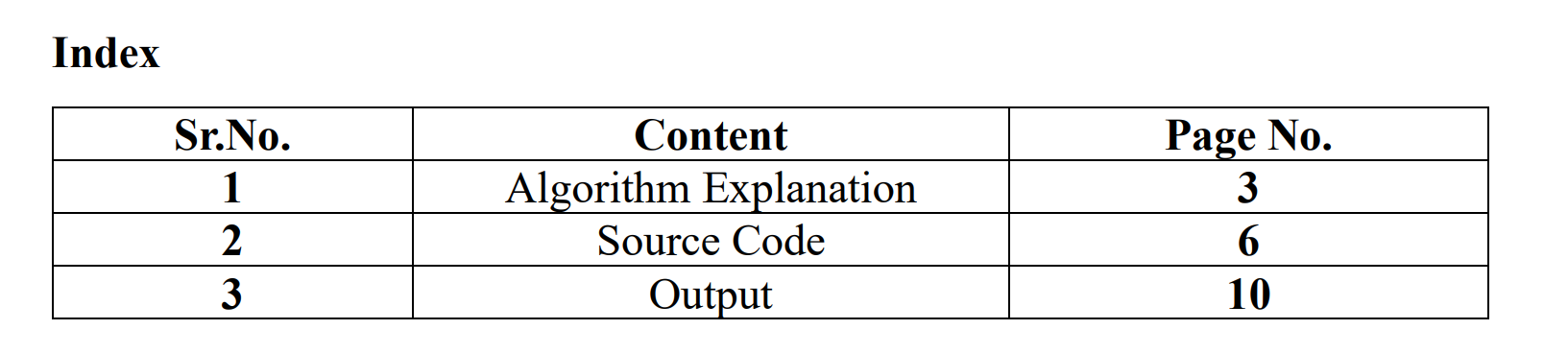
**2023-2024**

**Pursued in**

**Department of Computer Engineering**

**Faculty of Science & Technology**

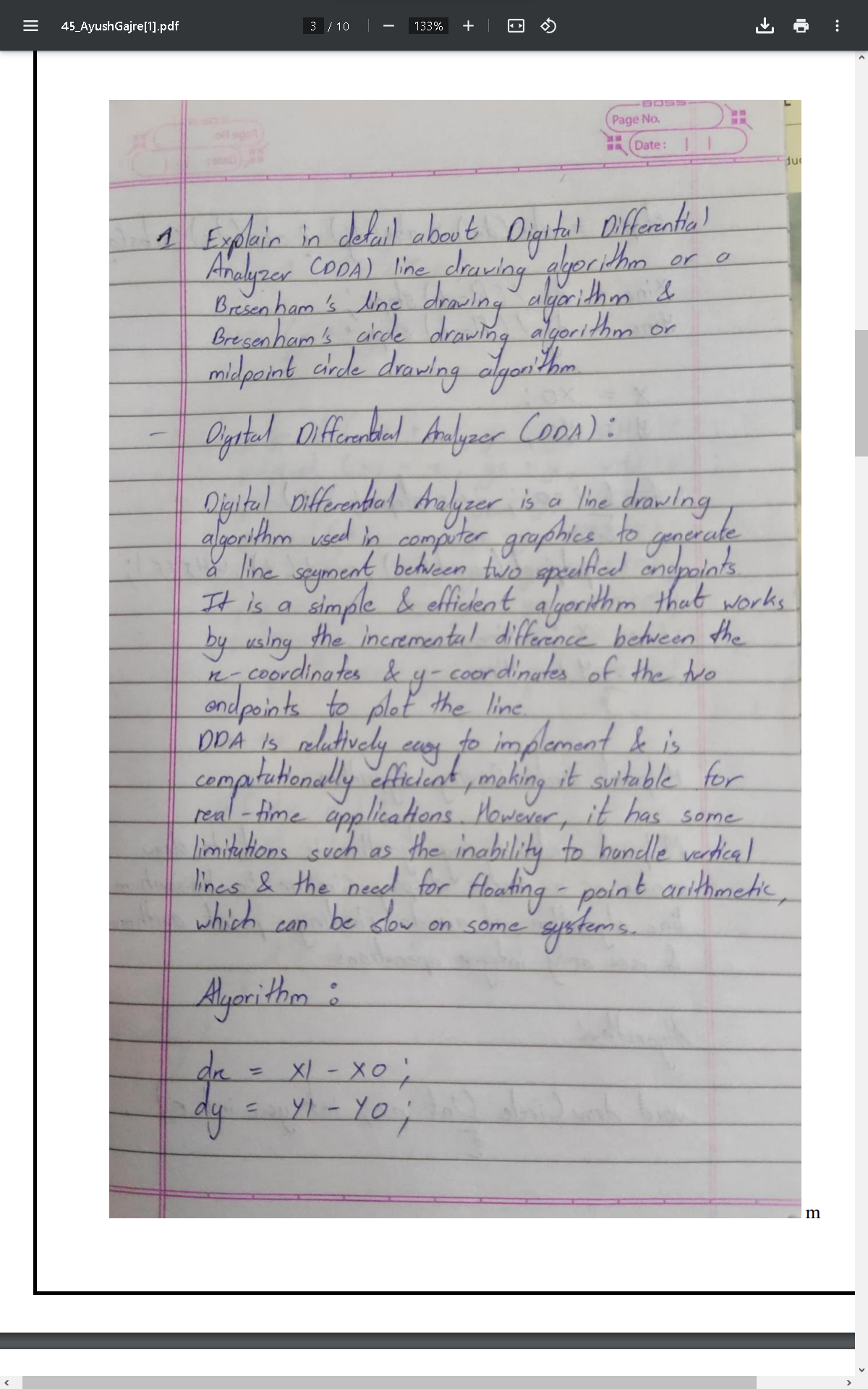
**Vishwakarma University, Pune**

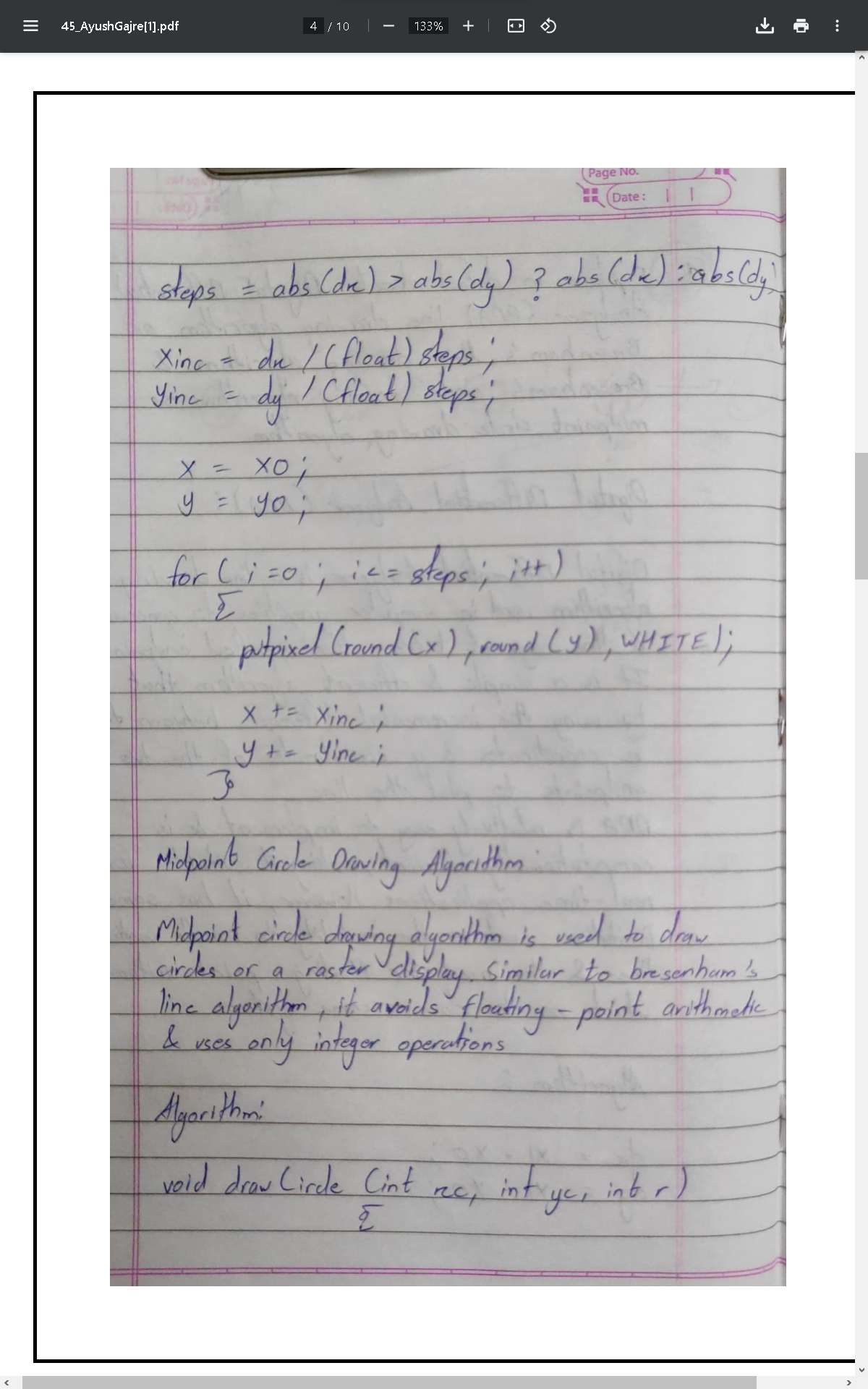
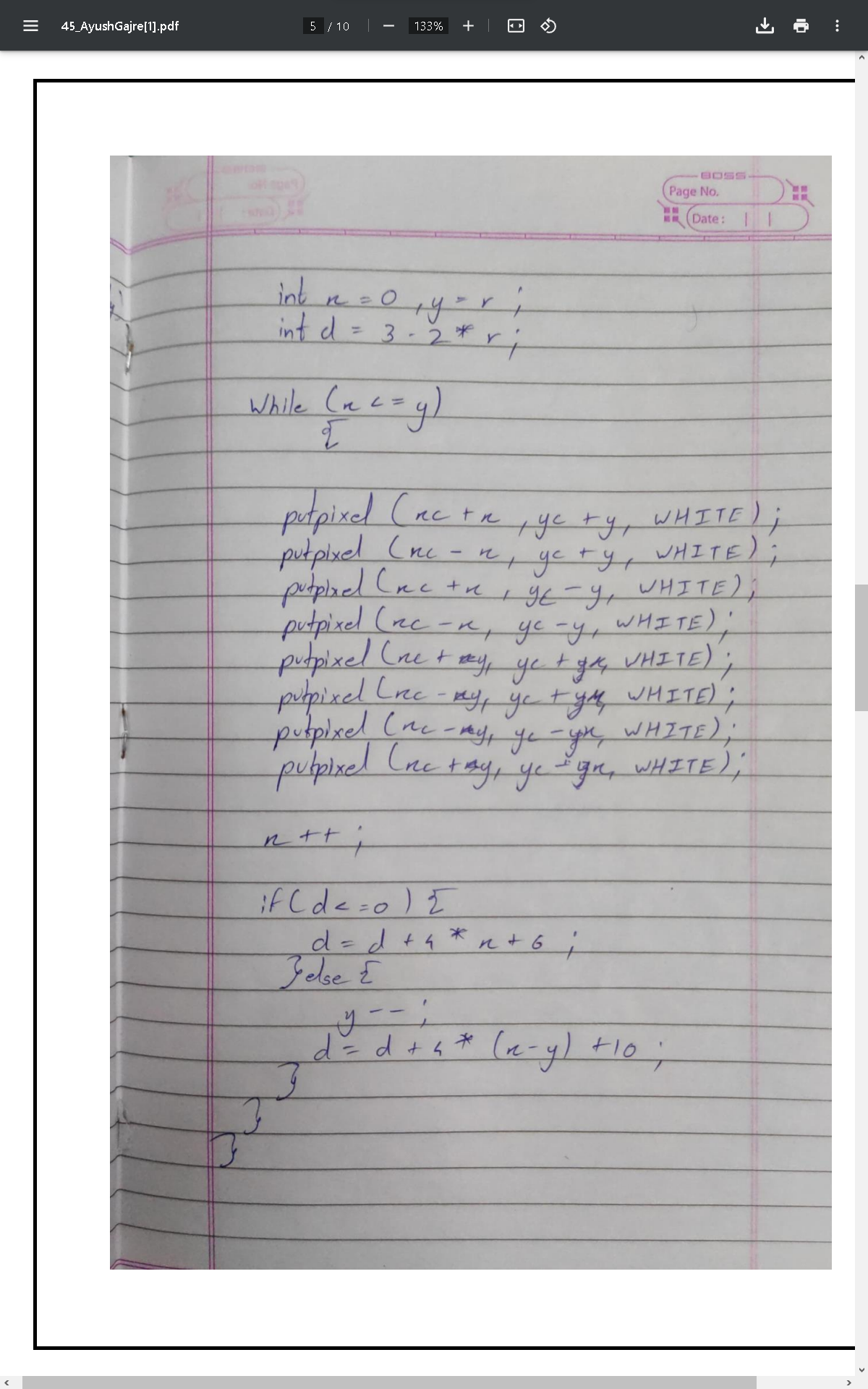
****

**Problem Statement:**

**The goal of this project is to experiment with different scan conversion algorithms. You will need to implement a Digital Differential Analyzer (DDA) line drawing algorithm or a Bresenham’s Line Drawing Algorithm and Bresenham’s Circle drawing algorithm or Midpoint Circle drawing algorithm. The rest of this document gives you the specific requirements as well as some ideas about how to proceed.**

**Explain in detail about Digital Differential Analyzer(DDA) line drawing algorithm or a Bresenham’s Line Drawing Algorithm and Bresenham’s Circle drawing algorithm or Midpoint Circle drawing algorithm**





**Source Code**

**#include <stdio.h>**

**#include <conio.h>**

**#include <graphics.h>**

**#include <math.h>**

**void dda(int x, int y, int x1, int y1) {**

**float m, n, steps, x2, y2, i;**

**m = (y1 - y);**

**n = (x1 - x);**

**if (fabs(n) >= fabs(m)) {**

**steps = fabs(n);**

**} else {**

**steps = fabs(m);**

**}**

**n = n / steps;**

**m = m / steps;**

**x2 = x;**

**y2 = y;**

**i = 1;**

**while (i <= steps) {**

**putpixel(x2, y2, WHITE);**

**x2 += n;**

**y2 += m;**

**i = i + 1;**

**}**

**}**

**void bresnln(int x1, int y1, int x2, int y2) {**

**int dx = abs(x2 - x1);**

**int dy = abs(y2 - y1);**

**int x, y;**

**int p = 2 \* dy - dx;**

**int twoDy = 2 \* dy, twoDyDx = 2 \* (dy - dx);**

**int xEnd;**

**if (x1 > x2) {**

**x = x2;**

**y = y2;**

**xEnd = x1;**

**} else {**

**x = x1;**

**y = y1;**

**xEnd = x2;**

**}**

**putpixel(x, y, WHITE);**

**while (x < xEnd) {**

**x++;**

**if (p < 0)**

**p += twoDy;**

**else {**

**y++;**

**p += twoDyDx;**

**}**

**putpixel(x, y, WHITE);**

**}**

**}**

**void cir(int xc, int yc, int r) {**

**int x = 0, y = r;**

**int p = 1 - r;**

**while (x <= y) {**

**putpixel(xc + x, yc + y, WHITE);**

**putpixel(xc - x, yc + y, WHITE);**

**putpixel(xc + x, yc - y, WHITE);**

**putpixel(xc - x, yc - y, WHITE);**

**putpixel(xc + y, yc + x, WHITE);**

**putpixel(xc - y, yc + x, WHITE);**

**putpixel(xc + y, yc - x, WHITE);**

**putpixel(xc - y, yc - x, WHITE);**

**x++;**

**if (p < 0)**

**p += 2 \* x + 1;**

**else {**

**y--;**

**p += 2 \* (x - y) + 1;**

**}**

**}**

**}**

**void midptcircle(int xc, int yc, int r) {**

**int x = 0, y = r;**

**int p = 1 - r;**

**while (x <= y) {**

**putpixel(xc + x, yc + y, WHITE);**

**putpixel(xc - x, yc + y, WHITE);**

**putpixel(xc + x, yc - y, WHITE);**

**putpixel(xc - x, yc - y, WHITE);**

**putpixel(xc + y, yc + x, WHITE);**

**putpixel(xc - y, yc + x, WHITE);**

**putpixel(xc + y, yc - x, WHITE);**

**putpixel(xc - y, yc - x, WHITE);**

**x++;**

**if (p < 0)**

**p += 2 \* x + 1;**

**else {**

**y--;**

**p += 2 \* (x - y) + 1;**

**}**

**}**

**}**

**int main() {**

**int i;**

**int gd = DETECT, gm;**

**initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");**

**circle(200, 300, 30);**

**circle(350, 300, 30);**

**arc(200, 300, 0, 180, 34);**

**arc(350, 300, 30, 195, 34);**

**dda(200, 300, 270, 300);**

**dda(200, 240, 200, 300);**

**dda(200, 250, 270, 300);**

**dda(270, 300, 350, 250);**

**dda(200, 250, 350, 250);**

**line(350, 300, 350, 220);**

**line(340, 210, 360, 230);**

**line(265, 280, 275, 320);**

**line(190, 240, 220, 240);**

**midptcircle(580, 100, 50);**

**line(0, 330, 650, 330);**

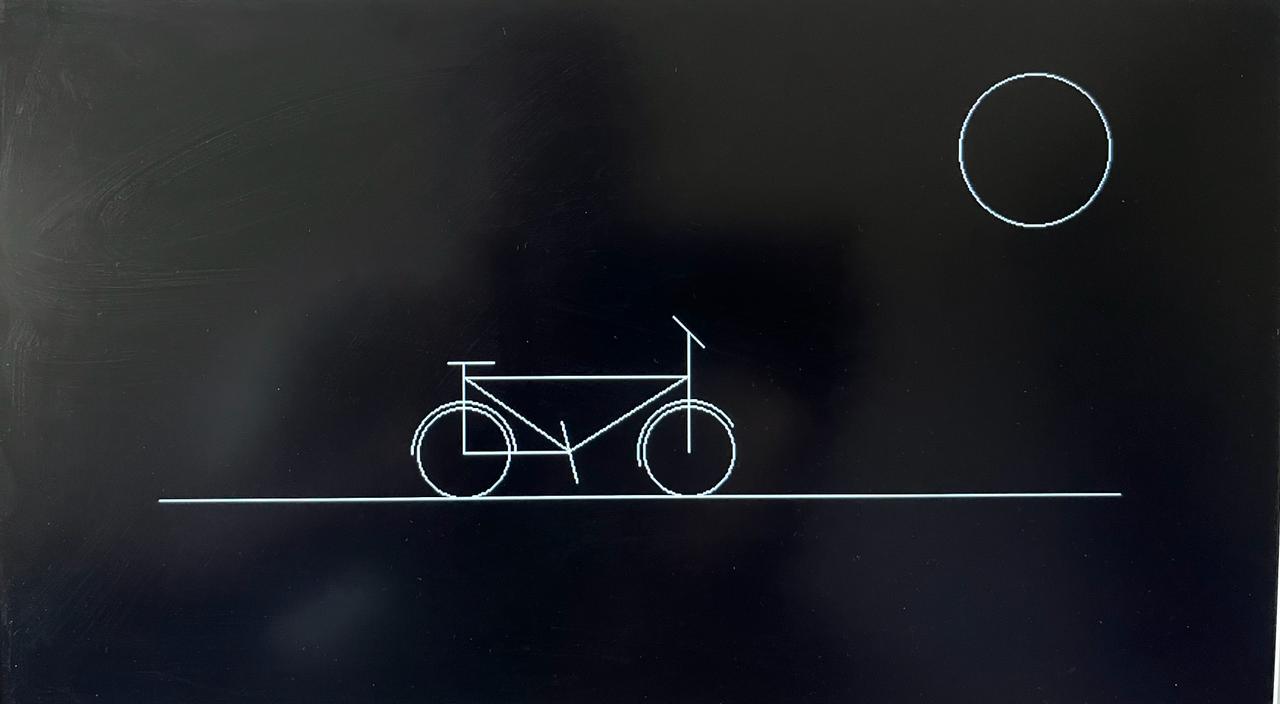
**getch();**

**closegraph();**

**return 0;**

**}**

**Output**

****

**Conclusion**

**The conclusion of the program appears to be the completion of a graphics display using the Turbo C graphics library. The program draws several shapes, including circles, arcs, lines, and polygons, to create an image of a cartoon character or object.**

**The shapes drawn include two circles representing the eyes, arcs for the eyebrows, lines for the mouth and other facial features, and a circle with lines representing a sun. The character or object depicted is not explicitly mentioned in the code, but it could be inferred based on the shapes and their arrangement.**

**Overall, the program demonstrates the usage of various graphics functions like circle, arc, line, dda, and midptcircle to draw shapes and create a graphical representation.**