Signa Institute of Professional Studies,Kanpur

[SUB - Computer Laboratory and Practical Work of Computer Graphics]

[BCA - 4001P]

Submitted To

Ms. Sameeksha Yadav Mam

Submitted By

Mohd Uvaish

Roll no: 22015002118

DATE- _____

1. Write a program to implement line.

```
#include <graphics.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int y_position = 200;
    line(0, y_position, getmaxx(), y_position);
    getch();
    closegraph();
}
```

2. Write a program to implement circle.

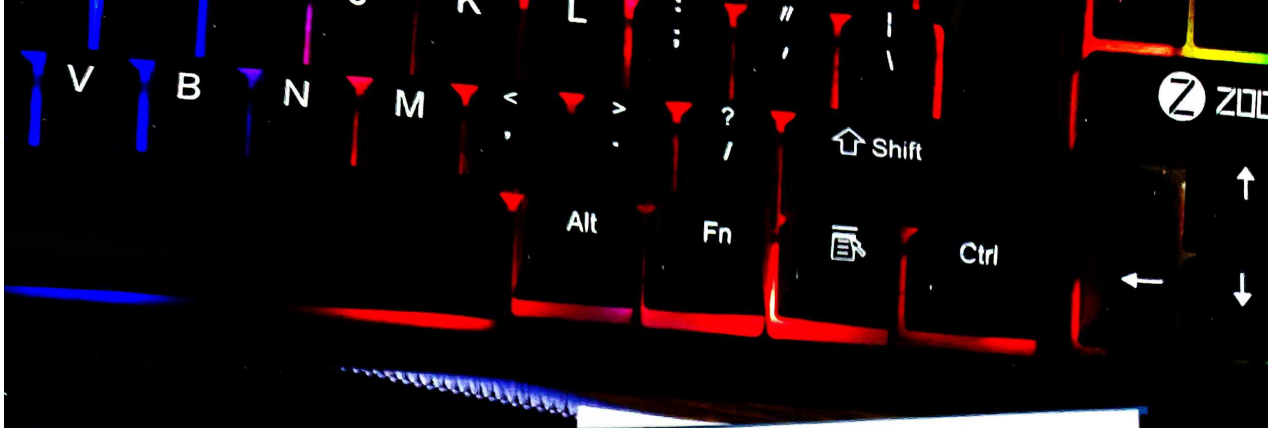
```
#include <graphics.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x_center = getmaxx()/2;
    int y_center = getmaxy()/2;

    circle(x_center, y_center, radius);

    getch();
    closegraph();
    return 0;
}
```

3. Write a program to draw 3D Sphere.

```
#include <graphics.h>
#include <math.h>

void draw3DSphere (int x_center, int y_center,
int radius) {
    int angle;
    float x,y, radian;
    int color_intensity;

    for (angle=0; angle
    <=180; angle += 1) {
        radian = angle * (M_PI / 180.0);
        y = radius * cos(radian);
        x = radius * sin(radian);

        float x_scaled = x * 1.6;

        color_intensity = 8 + (int) (7 * cos(radian));
        setcolor (color_intensity);
        setfillstyle (SOLID_FILL, color_intensity);

        fillellipse (x_center, y_center - y, x_scaled, x);
        fillellipse (x_center, y_center + y, x_scaled, x);
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph (&gd, &gm, "");
    int x_center = getmaxx()/2;
    int y_center = getmaxy()/2;
    int radius = 100;
    draw3DSphere (x_center, y_center, radius);
    getch();
    closegraph();
}
```

4. Write a program to draw 3D cube.

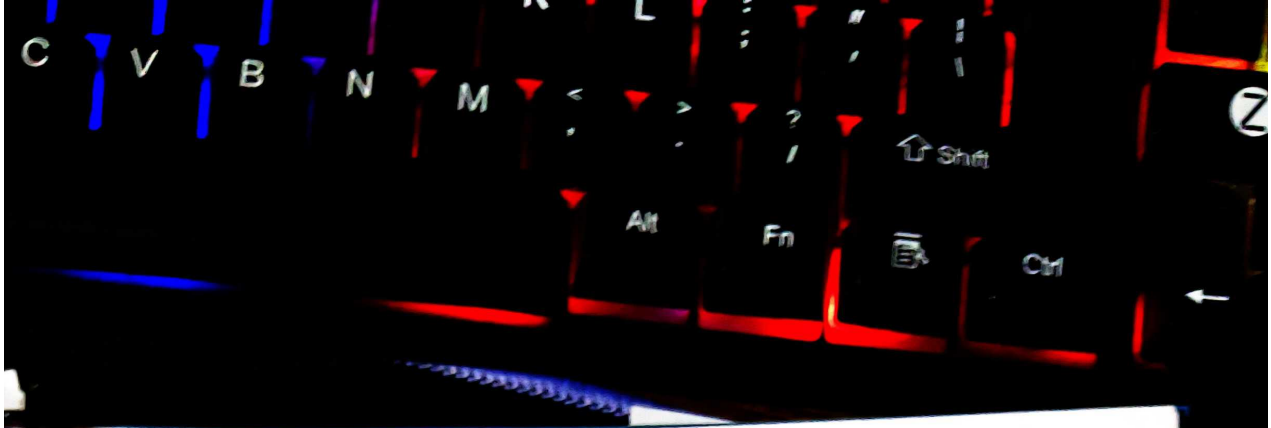
```
#include <graphics.h>
#include <stdlib.h>

typedef struct {
    int x, y, z;
} Point3D;

void project3Dto2D(Point3D p, int *x, int *y) {
    int fov = 256;
    int viewer_distance = 4;
    *x = p.x * fov / (p.z + viewer_distance + fov);
    *y = p.y * fov / (p.z + viewer_distance + fov);
}

void draw3DLine(Point3D p1, Point3D p2) {
    int x1, y1, x2, y2;
    project3Dto2D(p1, &x1, &y1);
    project3Dto2D(p2, &x2, &y2);
    line(x1 + getmaxx()/2, y1 + getmaxy()/2, x2 +
    getmaxx()/2, y2 + getmaxy()/2);
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    Point3D points[8] = {
        {-50, -50, -50}, {50, -50, -50},
        {50, 50, -50}, {-50, 50, -50},
```



```
        {-50,-50,50}, {50,-50,50},
        { 50, 50, 50}, {-50, 50, 50}
    };

    drawline(points[0], points[1]);
    drawline(points[1], points[2]);
    drawline(points[2], points[3]);
    drawline(points[3], points[0]);

    drawline(points[4], points[5]);
    drawline(points[5], points[6]);
    drawline(points[6], points[7]);
    drawline(points[7], points[4]);

    drawline(points[0], points[4]);
    drawline(points[1], points[5]);
    drawline(points[2], points[6]);
    drawline(points[3], points[7]);

    getch();
    closegraph();
}
```

5. Write a program to perform window to viewport transformation.

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

void windowToViewport (float wx, float wy, float
*vx, float *vy, float Wxmin, float Wxmax,
float Wymin, float Wymax, int Vxmin, int Vxmax,
int Vymin, int Vymax) {

    float sx = (Vxmax - Vxmin) / (Wxmax - Wxmin);
    float sy = (Vymax - Vymin) / (Wymax -
    Wymin);

    *vx = Vxmin + (wx - Wxmin) * sx;
    *vy = Vymin + (wy - Wymin) * sy;
}

int main () {
    int gd = DETECT, gm;
    initgraph (&gd, &gm, NULL);

    float Wxmin = -100, Wxmax = 100, Wymin =
    -100, Wymax = 100;
    int Vxmin = 100, Vxmax = 500, Vymin = 100,
    Vymax = 600;
    float wx = 30, wy = 30;
    float vx, vy;
```



```
windowToViewport (wx, wy, Bvx, Bvy, Wxmin,
Wxmax, Wymin, Wymax, Vxmin, Vxmax, Vymin,
Vymax);

putpixel ((int)vx, (int)vy, WHITE);
rectangle (Vxmin, Vymin, Vxmax, Vymax);

getch();
closegraph();
}
```

6. Write a program to draw 2D shape like triangle.

```
#include <graphics.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    int x1 = 100, y1 = 200;
    int x2 = 200, y2 = 100;
    int x3 = 300, y3 = 200;

    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);

    getch();
    closegraph();
}
```

7. Write a program to draw 2D shape like rectangle

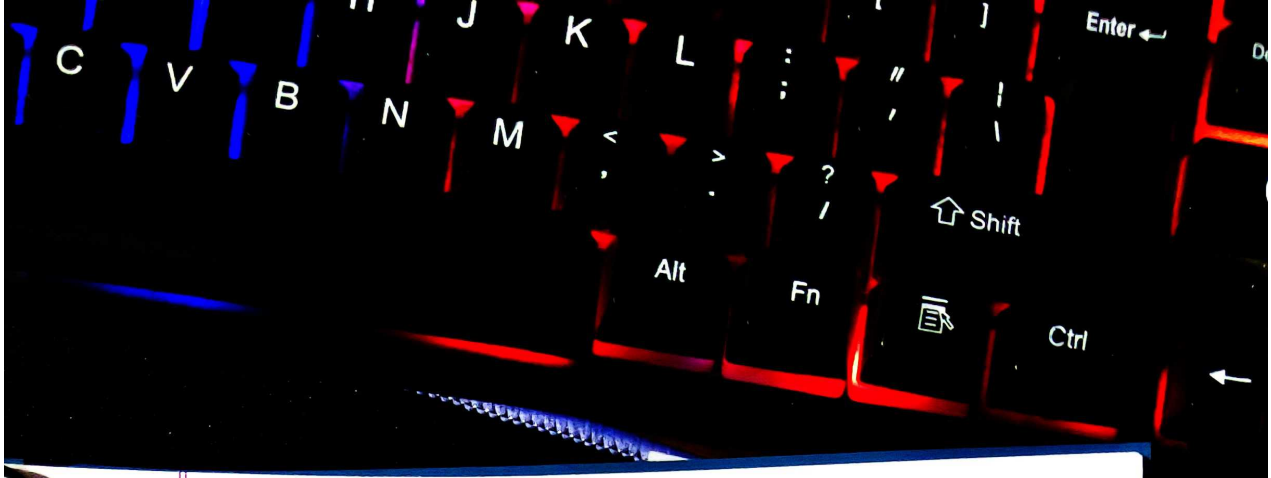
```
#include <graphics.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    int left = 100;
    int right = 100;

    int right = 300;
    int bottom = 200;

    rectangle(left, top, right, bottom);
    getch();
    closegraph();
}
```

8. Write a program to draw basic geometric pattern like sphere.

```
#include <graphics.h>
#include <math.h>

void drawGradientCircle(int x_center, int y_center,
int radius) {
    int color_intensity;
    for(int r = radius; r > 0; r -= 2) {

        color_intensity = 15 - (15 * r / radius);
        setcolor(color_intensity);
        circle(x_center, y_center, r);
        setfillstyle(SOLID_FILL, color_intensity);
        floodfill(x_center, y_center, color_intensity);
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    int x_center = getmaxx()/2;
    int y_center = getmaxy()/2;
    int radius = 100;

    drawGradientCircle(x_center, y_center, radius);
    getch();
    closegraph();
}
```

9. Write a program to draw line using Bresenham's algorithm

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

void drawVerticalLineBresenham(int x, int y0,
int y1) {
    int dy = abs(y1 - y0), sy = y0 < y1 ? 1 : -1;
    int err = dy / 2, y;

    for (y = y0, y != y1; y += sy) {
        putpixel(x, y, WHITE);
        err -= dy;
        if (err < 0) {
            err += dy;
        }
    }
    putpixel(x, y1, WHITE);
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    int x = 200;
    int y0 = 50, y1 = 300;
    drawVerticalLineBresenham(x, y0, y1);
    getch()
```



Date
Page

```
closegraph();
}
```

10. Write program to draw eclipse using Bresenham's Algorithm

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

void drawEllipse(int xc, int yc, int width, int height) {
    int a2 = width * width;
    int b2 = height * height;
    int fa2 = 4 * a2, fb2 = 4 * b2;
    int x, y, sigma;

    for (x=0, y=height, sigma = 2*b2+a2*(1-2*height); b2*x <= a2*y; x++) {
        putpixel(xc+x, yc+y, WHITE);
        putpixel(xc-x, yc+y, WHITE);
        putpixel(xc+x, yc-y, WHITE);
        putpixel(xc-x, yc-y, WHITE);
        if (sigma >=0) {
            sigma += fa2 * (1-y);
            y--;
        }
        sigma += b2 * ((4*x)+6);
    }

    for(x = width, y=0, sigma = 2*a2+b2*(1-2*width)
    ; a2*y <= b2*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);
        if (sigma >=0){
            sigma += fb2 * (1-x);
            x--;
        }
        sigma += a2 * ((4*y)+6);
    }
}

int main(){
    int gd = DETECT, gm;
    int xc = 320, yc = 240, width = 100, height = 50;
    initgraph(&gd, &gm, NULL);
    drawEllipse(xc, yc, width, height);
    getch();
    closegraph();
}
```

11. Write a program to draw 2D Smiley face using circle & line.

```
#include <graphics.h>
#include <conio.h>

int main()
{
    int gd = DETECT, gm;
    int x, y, radius;
    initgraph(&gd, &gm, NULL);
    x = getmaxx() / 2;
    y = getmaxy() / 2;
    radius = 200;

    setcolor(YELLOW);
    setfillstyle(SOLID_FILL, YELLOW);
    fillellipse(x, y, radius, radius);

    setcolor(BLACK);
    setfillstyle(SOLID_FILL, BLACK);
    fillellipse(x - radius/3, y - radius/3, radius/10,
    radius/10);
    fillellipse(x + radius/3, y - radius/3, radius/10,
    radius/10);

    setcolor(BLACK);
    arc(x, y, 200, 340, radius/2);
    getch();
    closegraph();
}
```

12. Write a program to rotate 2D shape about origin.

```
#include <graphics.h>
#include <conio.h>

void drawAndReflectRectangle(int left, int top,
int right, int bottom) {
        rectangle(left, top, right, bottom);
        rectangle(-right, -bottom, -left, -top);
}
int main() {
    int gd=DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int midx = getmaxx()/2;
    int midy = getmaxy()/2;

    int left = midx - 50, top = midy - 30, right =
    midx + 50, bottom = midy + 30;
    setcolor(YELLOW);
    setviewport(midx, midy, getmaxx(), getmaxy(),
    1);
    line(-getmaxx()/2, 0, getmaxx()/2, 0);
    line(0, -getmaxy()/2, 0, getmaxy()/2);
    drawAndReflectRectangle(left - midx, top - midy,
    right - midx, bottom - midy);
    getch();
    closegraph();
}
```

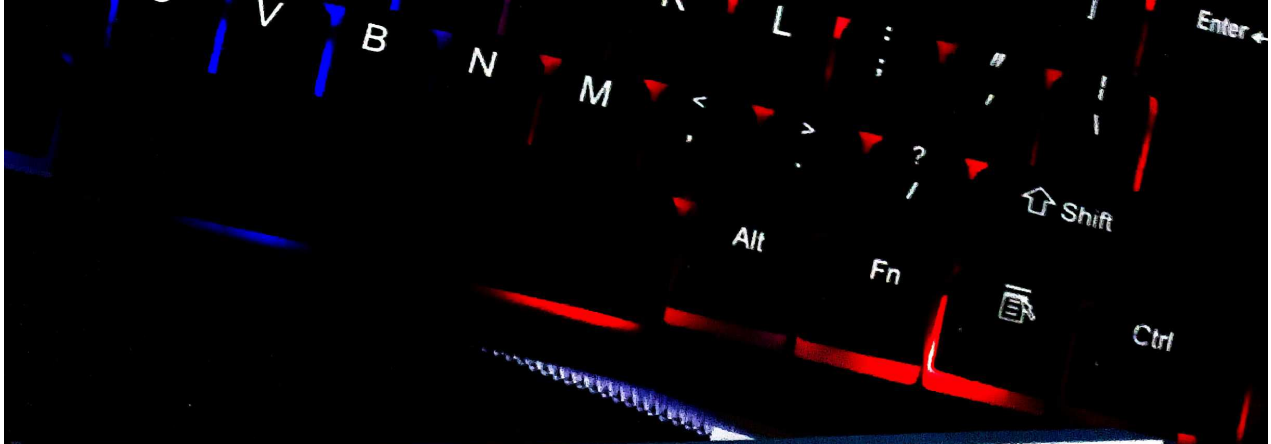
13. Implement scaling transformation for 2D shape.

```
#include <graphics.h>
#include <conio.h>

void drawScaledRectangle(int left, int top,
int right, int bottom, float scaleX, float scaleY)
{
    rectangle(Left, top, right, bottom);

    int centerX = (Left + right) / 2;
    int centerY = (top + bottom) / 2;

    int newLeft = centerX + (int)((Left - centerX)
    * scaleX);
    int newTop = centerY + (int)((top - centerY)
    * scaleY);
    int newRight = centerX + (int)((right - centerX)
    * scaleX);
    int newBottom = centerY + (int)((bottom -
    centerY) * scaleY);

    rectangle(newLeft, newTop, newRight, newBottom
    );
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
```

```
int left=200, itop = 150, iright = 300,
bottom = 250;

float scaleX = 1.5, scaleY = 1.5;
setcolor (~~GREEN~~);
          RED
drawScaledRectangle (left, top, right, bottom,
1.0, 1.0);

setcolor (GREEN);
drawScaledRectangle (left, top, right, bottom,
scale X, scaleY);
getch();
closegraph();
}
```

14. Implement cohen-sutherland line clipping algorithm.

```
#include <graphics.h>
#include <conio.h>

const int xMin = 100, yMin = 100, xMax = 300,
yMax = 200;

int computeOutcode (double x, double y) {
    int code = 0;
    if (y > yMax) { code |= 1; }
    if (y < yMin) { code |= 2; }
    if (x > xMax) { code |= 4; }
    if (x < xMin) { cod |= 8; }
    return code;
}

void cohenSutherlandClip (double x0, double y0,
double x1, double y1) {
    int outcode0 = computeOutcode (x0, y0);
    int outcode1 = computeOutcode (x1, y1);
    bool accept = false;

    while (true) {
        if (!(outcode0 | outcode1)) {
            accept = true;
            break;
        } else if (outcode0 & outcode1) {
            break;
        } else {
```

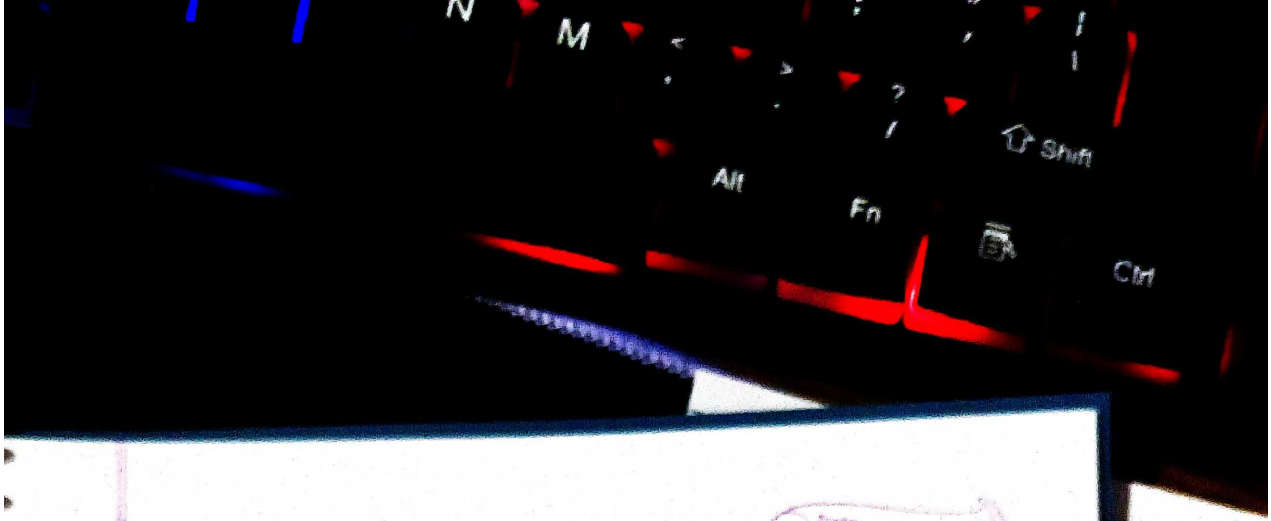


```
double x, y;
int outcodeOut = outcode0 ? : outcode0 :
outcode1;
if (outcodeOut & 1) {
    x = x0 + (x1-x0) * (yMax - y0) /
    (y1 - y0);
    y = yMax;
} else if (outcodeOut & 2) {
    x = x0 + (x1 - x0) * (yMin - y0) /
    (y1 - y0);
    y = yMin;
} else if (outcodeout & 4) {
    y = y0 + (y1-y0) * (xMax - x0) /
    (x1 - x0);
    x = xMax;
} else {
    y = y0 + (y1 - y0) * (xMin - x0) /
    (x1 - x0);
    x = xMin;
}

if (outcodeOut == outcode0) {
    x0 = x;
    y0 = y;
    outcode 0 = computeOutcode(x0, y0);
} else {
    x1 = x;
    y1 = y;
    outcode1 = computeOutcode (x1, y1);
}
}
}
```

```
    if (accept) {
        line (x0, y0, x1, y1);
    }
}

int main () {
    int gd = DETECT, gm;
    initgraph (&gd, &gm, NULL);

    rectangle (xMin, yMin, xMax, yMax);

    setcolor (RED);
    line (50, 50, 350, 250);

    setcolor (GREEN);
    cohenSutherlandClip (50, 50, 350, 250);

    getch();
    closegraph ();
}
```

15. Implementing basic drawing tool with functionalities like line drawing.

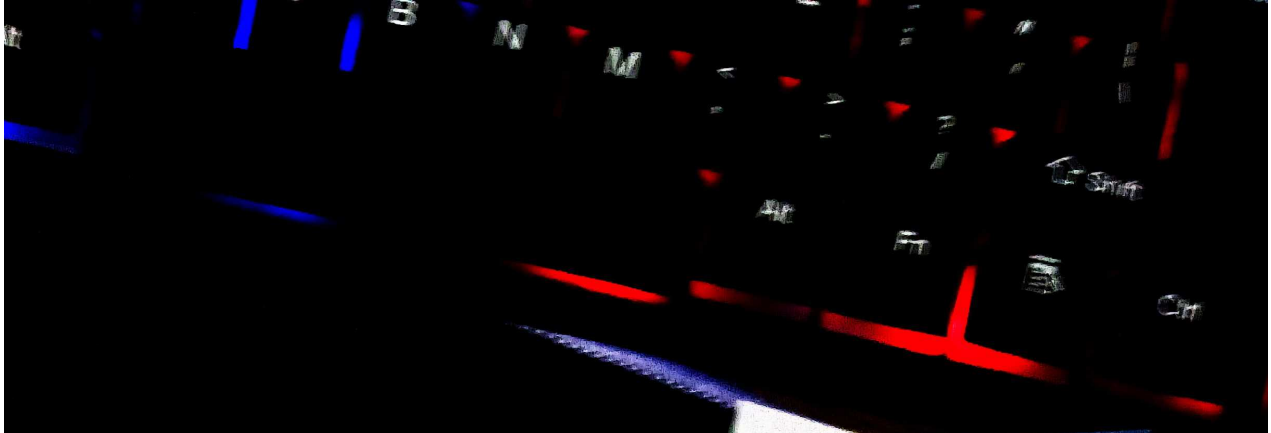
```
#include <SDL.h>
#include <iostream>

int main(int argc, char* argv[]) {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window = SDL_CreateWindow(
    "Simple Drawing Tool", SDL_WINDOWPO
    S_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
    640, 480, 0);

    SDL_Renderer* renderer = SDL_CreateRenderer(
    window, -1, SDL_RENDERER_ACCELERATED);

    bool quit = false;
    SDL_Event event;
    bool drawing = false;
    int startX = 0, startY = 0, endX = 0, endY = 0;

    while (!quit) {
        while (SDL_PollEvent(&event)) {
            switch (event.type) {
                case SDL_QUIT:
                quit = true;
                break;
                case SDL_MOUSEBUTTONDOWN:
                drawing = true;
                startX = event.button.x;
```



```
            startY = event.button.y;
            break;
        case SDL_MOUSEBUTTONUP:
            drawing = false;
            endX = event.button.x;
            endY = event.button.y;

            SDL_SetRenderDrawColor(
            renderer, 255, 255, 255,
            SDL_ALPHA_OPAQUE);
            SDL_RenderDrawLine(renderer,
            startX, startY, endX, endY);
            SDL_RenderPresent(renderer);
            break;
        }
    }
}

SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();
}
```

16. Write a program to draw a circle using Mid point circle drawing Algorithm.

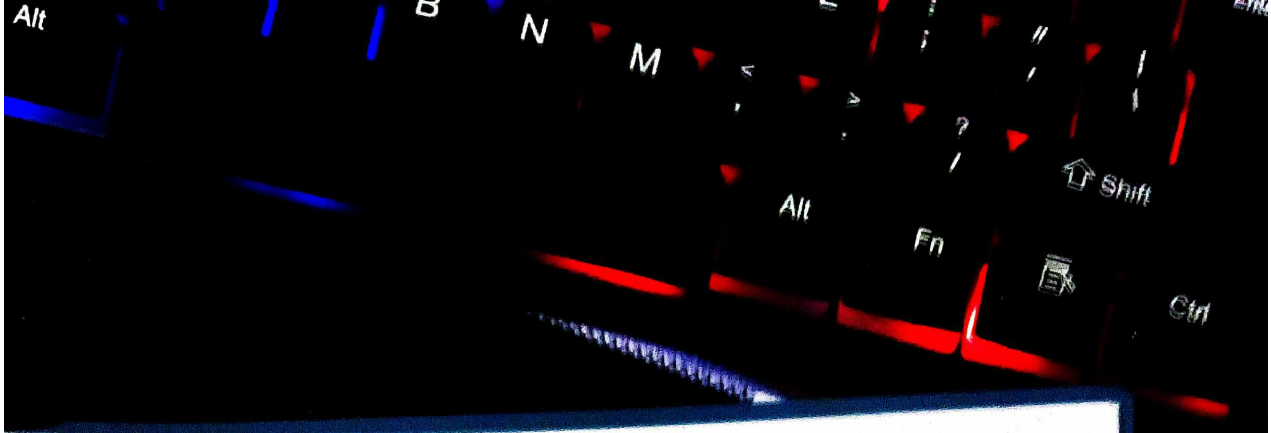
```
#include <graphics.h>
#include <conio.h>

void plotPoints(int xc, int yc, int x, int 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);
}

void midPointCircleDraw(int xc, int yc, int r){
    int x=0, y=r;
    int p=1-r;

    plotPoints(xc, yc, x, y);

    while(x<y){
        x++;
        if(p<0){
            p=p+2*x+1;
        }else{
            y--;
            p=p+2*(x-y)+1;
        }
```



17. Implement basic Image processing operations like scaling.

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

int main (int argc, char* argv[]) {
    if (SDL_Init(SDL_INIT_VIDEO) < 0) {
        printf("SDL could not initialise: SDL_
        Error: %s\n", SDL_GetError());
        return 1;
    }

    SDL_Window* window = SDL_CreateWindow("
    SDL Tutorial", SDL_WINDOWPOS_UNDEFINED,
    SDL_WINDOWPOS_UNDEFINED, 800, 600,
    SDL_WINDOW_SHOWN);
    if (window == NULL) {
        printf("Window could not be created!
        SDL_Error: %s\n", SDL_GetError());
        SDL_Quit();
        return 1;
    }

    SDL_Renderer* renderer = SDL_CreateRenderer(
    window, -1, SDL_RENDERER_ACCELERATED);
    SDL_Surface* loadedSurface = SDL_LoadBMP("
    example.bmp");
    if (loadedSurface == NULL) {
        printf("Unable to load image! SDL
```

```
        Error: %s\n", SDL_GetError());
        SDL_DestroyWindow(window);
        SDL_Quit();
        return 1;
    }

    SDL_Texture* originalTexture = SDL_CreateTexture
    FromSurface(renderer, loadedSurface);
    if (originalTexture == NULL) {
        printf("Unable to create texture from
        surface! SDL Error: %s\n", SDL_GetError
        ());
    }

    int scale = 2;
    SDL_Rect scaleRect = {loadedSurface->w, 0,
    loadedSurface->w * scale, loadedSurface->h *
    scale};
    SDL_Texture* scaledTexture = SDL_CreateTexture(
    renderer, SDL_PIXELFORMAT_RGBA8888, SDL_TEXT
    URE_ACCESS_TARGET, scaleRect.w, scaleRect.h);

    SDL_SetRenderTarget(renderer, scaledTexture);
    SDL_RenderCopy(renderer, originalTexture, NULL,
    &scaleRect);
    SDL_SetRenderTarget(renderer, NULL);

    SDL_RenderClear(renderer);

    SDL_RENDERCopy(render, originalTexture, NULL, NULL);
    SDL_RenderCopy(renderer, scaledTexture, NULL,
```



```
&ScaleRect);

SDL_RenderPresent(renderer);

SDL_Event e;
bool quit = false;
while(!quit)
    while (SDL_PollEvent(&e) != 0) {
        if (e.type == SDL_QUIT) {
            quit = true;
        }
    }
}

SDL_FreeSurface(loadedSurface);
SDL_DestroyTexture(originalTexture);
SDL_DestroyTexture(scaledTexture);
SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(window);
SDL_Quit();
}
```

1B. Write a program to implement vertical line.

```
#include <graphics.h>
#include <conio.h>

int main()
{
    int gd=DETECT, gm;
    int x, y0, y1;
    initgraph(&gd, &gm, NULL);

    x = getmaxx()/2;

    y0= 0;
    y1 = getmaxy();

    setcolor(WHITE);

    line(x, y0, x, y1);
    getch();
    closegraph();
}
```

19. Write a program to implement the Horizontal line.

```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    int x0, x1, y;
    initgraph(&gd, &gm, NULL);

    y = getmaxy() / 2;
    x0 = 0;
    x1 = getmaxx();

    setcolor(WHITE);

    line(x0, y, x1, y);

    getch();

    closegraph();
}
```

20. Write program to draw animation using increasing circle filled with colors & patterns.

```
#include <graphics.h>
#include <conio.h>
void main(){
    int gd = DETECT, gm, i, x, y;
    initgraph(&gd, &gm, "C:\TC\BGI");
    x = getmaxx()/3;
    y = getmaxy()/3;
    setcolor(WHITE);
    Setcolor(BLUE);
    for(i=1; i<=9; i++){
        setfillstyle(i,i);
        delay(20);
        circle(x,y,i*20);
        floodfill(x+i*20,y,BLUE);
    }
    getch();
    closegraph();
}
```

21. Write a program to draw basic construction [arc].

```
#include <graphics.h>
#include <conio.h>
void main() {
    int gd=DETECT, gm;
    initgraph(&gd, &gm, NULL);
    arc(120, 200, 180, 0, 30);
    getch();
    closegraph();
}
```

23 Write a program to draw the rectangle with background rectangle

```
#include <graphics.h>
void main()
{
    int gd = DETECT, gm;
    int left = 150, etop = 150;
    int right = 450, bottom = 450;
    initgraph (&gd, &gm, NULL);
    setfillstyle (SOLID_FILL, RED);
    rectangle (left, etop, right, bottom);
    getch();
    closegraph();
}
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

25. Write a program to set the Background color.

```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    setbkcolor(BLUE);
    cleardevice();
    outtextxy(200, 200, "Background color is now
    blue");

    getch();
    closegraph();
}
```

26. Write a program to set Background color of arc.

```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    int x = 300, y = 200;
    int start_angle = 0, end_angle = 180;
    int radius = 100;
    initgraph (&gd, &gm, NULL);

    setbkcolor (BLUE);
    cleardevice ();

    setcolor (RED);

    arc (x, y, start_angle, end_angle, radius);

    setcolor (WHITE);
    outtextxy (10, 10, "Press any key to exit...");

    getch();
    closegraph();
}
```

27. Write a program to draw different linestyles like dotted line, solid line, dashed line.

```
#include <graphics.h>

int main() {
    int gd = DETECT, gm;
    int c;
    int x = 200, y = 100;
    initgraph(&gd, &gm, NULL);

    for (c=0; c<5; c++) {
        setlinestyle(c, 0, 1);

        line(x, y, x+200, y);
        y = y+25;
    }

    getch();
    closegraph();
}
```