```
// C program to implement
// Cubic Bezier Curve

/* install SDL library for running thing code*/
/* install by using this commamnd line : sudo apt-get install libsdl2-dev */
/* run this code using command : gcc fileName.c -lSDL2 -lm*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<SDL2/SDL.h>

SDL_Window* window = NULL;
SDL_Renderer* renderer = NULL;
int mousePosX , mousePosY ;
int xnew , ynew ;

/*Function to draw all other 7 pixels present at symmetric position*/
void drawCircle(int xc, int yc, int x, int y)
{
    SDL_RenderDrawPoint(renderer,xc+x,yc+y) ;
    SDL_RenderDrawPoint(renderer,xc-x,yc+y);
    SDL_RenderDrawPoint(renderer,xc+x,yc-y);
    SDL_RenderDrawPoint(renderer,xc-x,yc-y);
    SDL_RenderDrawPoint(renderer,xc+y,yc+x);
    SDL_RenderDrawPoint(renderer,xc-y,yc+x);
    SDL_RenderDrawPoint(renderer,xc+y,yc-x);
    SDL_RenderDrawPoint(renderer,xc-y,yc-x);
}

/*Function for circle-generation using Bresenham's algorithm */
void circleBres(int xc, int yc, int r)
{
    int x = 0, y = r;
    int d = 3 - 2 * r;
    while (y >= x)
    {
        /*for each pixel we will draw all eight pixels */
        drawCircle(xc, yc, x, y);
        x++;

        /*check for decision parameter and correspondingly update d, x, y*/
        if (d > 0)
        {
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else
            d = d + 4 * x + 6;
        drawCircle(xc, yc, x, y);
    }
}

/* Function that take input as Control Point x_coordinates and
Control Point y_coordinates and draw bezier curve */
void bezierCurve(int x[] , int y[])
{
    double xu = 0.0 , yu = 0.0 , u = 0.0 ;
    int i = 0 ;
    for(u = 0.0 ; u <= 1.0 ; u += 0.0001)
    {
        xu = pow(1-u,3)*x[0]+3*u*pow(1-u,2)*x[1]+3*pow(u,2)*(1-u)*x[2]
            +pow(u,3)*x[3];
        yu = pow(1-u,3)*y[0]+3*u*pow(1-u,2)*y[1]+3*pow(u,2)*(1-u)*y[2]
            +pow(u,3)*y[3];
        SDL_RenderDrawPoint(renderer , (int)xu , (int)yu) ;
    }
}
int main(int argc, char* argv[])
{
    /*initialize sdl*/
    if (SDL_Init(SDL_INIT_EVERYTHING) == 0)
    {
        /*
            This function is used to create a window and default renderer.
            int SDL_CreateWindowAndRenderer(int width
                                    ,int height
                                    ,Uint32 window_flags
                                    ,SDL_Window** window
                                    ,SDL_Renderer** renderer)
            return 0 on success and -1 on error
        */
        if(SDL_CreateWindowAndRenderer(640, 480, 0, &window, &renderer) == 0)
        {
            SDL_bool done = SDL_FALSE;

            int i = 0 ;
            int x[4] , y[4] , flagDrawn = 0 ;

            while (!done)
            {
                SDL_Event event;

                /*set background color to black*/
                SDL_SetRenderDrawColor(renderer, 0, 0, 0, SDL_ALPHA_OPAQUE);
                SDL_RenderClear(renderer);

                /*set draw color to white*/
                SDL_SetRenderDrawColor(renderer, 255, 255, 255, SDL_ALPHA_OPAQUE);

                /* We are drawing cubic bezier curve
                which has four control points */
                if(i==4)
                {
                    bezierCurve(x , y) ;
                    flagDrawn = 1 ;
                }

                /*grey color circle to encircle control Point P0*/
                SDL_SetRenderDrawColor(renderer, 128, 128, 128, SDL_ALPHA_OPAQUE);
                circleBres(x[0] , y[0] , 8) ;

                /*Red Line between control Point P0 & P1*/
                SDL_SetRenderDrawColor(renderer, 255, 0, 0, SDL_ALPHA_OPAQUE);
                SDL_RenderDrawLine(renderer , x[0] , y[0] , x[1] , y[1]) ;

                /*grey color circle to encircle control Point P1*/
                SDL_SetRenderDrawColor(renderer, 128, 128, 128, SDL_ALPHA_OPAQUE);
                circleBres(x[1] , y[1] , 8) ;

                /*Red Line between control Point P1 & P2*/
                SDL_SetRenderDrawColor(renderer, 255, 0, 0, SDL_ALPHA_OPAQUE);
                SDL_RenderDrawLine(renderer , x[1] , y[1] , x[2] , y[2]) ;

                /*grey color circle to encircle control Point P2*/
                SDL_SetRenderDrawColor(renderer, 128, 128, 128, SDL_ALPHA_OPAQUE);
                circleBres(x[2] , y[2] , 8) ;

                /*Red Line between control Point P2 & P3*/
                SDL_SetRenderDrawColor(renderer, 255, 0, 0, SDL_ALPHA_OPAQUE);
                SDL_RenderDrawLine(renderer , x[2] , y[2] , x[3] , y[3]) ;
                /*grey color circle to encircle control Point P3*/
                SDL_SetRenderDrawColor(renderer, 128, 128, 128, SDL_ALPHA_OPAQUE);
                circleBres(x[3] , y[3] , 8) ;

                /*We are Polling SDL events*/
                if (SDL_PollEvent(&event))
                {
                    /* if window cross button clicked then quit from window */
                    if (event.type == SDL_QUIT)
                    {
                        done = SDL_TRUE;
                    }
                    /*Mouse Button is Down */
                    if(event.type == SDL_MOUSEBUTTONDOWN)
                    {
                        /*If left mouse button down then store
                        that point as control point*/
                        if(event.button.button == SDL_BUTTON_LEFT)
                        {
                            /*store only four points
                            because of cubic bezier curve*/
                            if(i < 4)
                            {
                                printf("Control Point(P%d):(%d,%d)\n"
                                ,i,mousePosX,mousePosY) ;

                                /*Storing Mouse x and y positions
                                in our x and y coordinate array */
                                x[i] = mousePosX ;
                                y[i] = mousePosY ;
                                i++ ;
                            }
                        }
                    }
                    /*Mouse is in motion*/
                    if(event.type == SDL_MOUSEMOTION)
                    {
                        /*get x and y positions from motion of mouse*/
                        xnew = event.motion.x ;
                        ynew = event.motion.y ;

                        int j ;

                        /* change coordinates of control point
                        after bezier curve has been drawn */
                        if(flagDrawn == 1)
                        {
                            for(j = 0 ; j < i ; j++)
                            {
                                /*Check mouse position if in b/w circle then
                        change position of that control point to mouse new
                                position which are coming from mouse motion*/
                                if((float)sqrt(abs(xnew-x[j]) * abs(xnew-x[j])
                                    + abs(ynew-y[j]) * abs(ynew-y[j])) < 8.0)
                                {
                                    /*change coordinate of jth control point*/
                                    x[j] = xnew ;
                                    y[j] = ynew ;
                                    printf("Changed Control Point(P%d):(%d,%d)\n"
                                        ,j,xnew,ynew) ;
                                }
                            }
                        }
                        /*updating mouse positions to positions
                        coming from motion*/
                        mousePosX = xnew ;
                        mousePosY = ynew ;
                    }
                }
                /*show the window*/
                SDL_RenderPresent(renderer);
            }
        }
        /*Destroy the renderer and window*/
        if (renderer)
        {
            SDL_DestroyRenderer(renderer);
        }
        if (window)
        {
            SDL_DestroyWindow(window);
        }
    }
    /*clean up SDL*/
    SDL_Quit();
    return 0;
}
```