Name : Vaishnavi Bharat Gangurde
Roll No : 25
Division : A

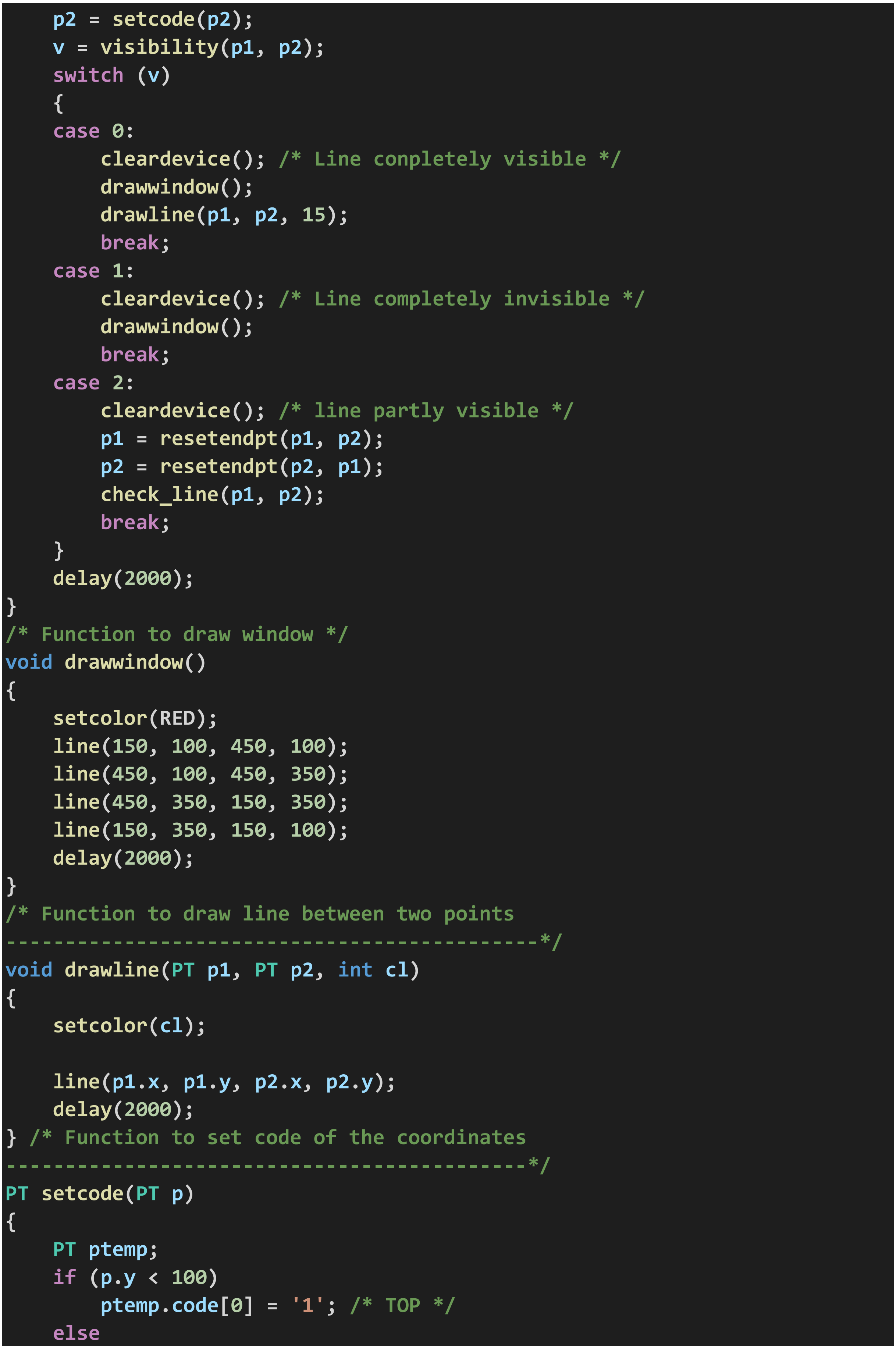
Assignment No : 2_2

Problem Statement :

Write a c++ program to implement cohen Sutherland line clipping algorithm.

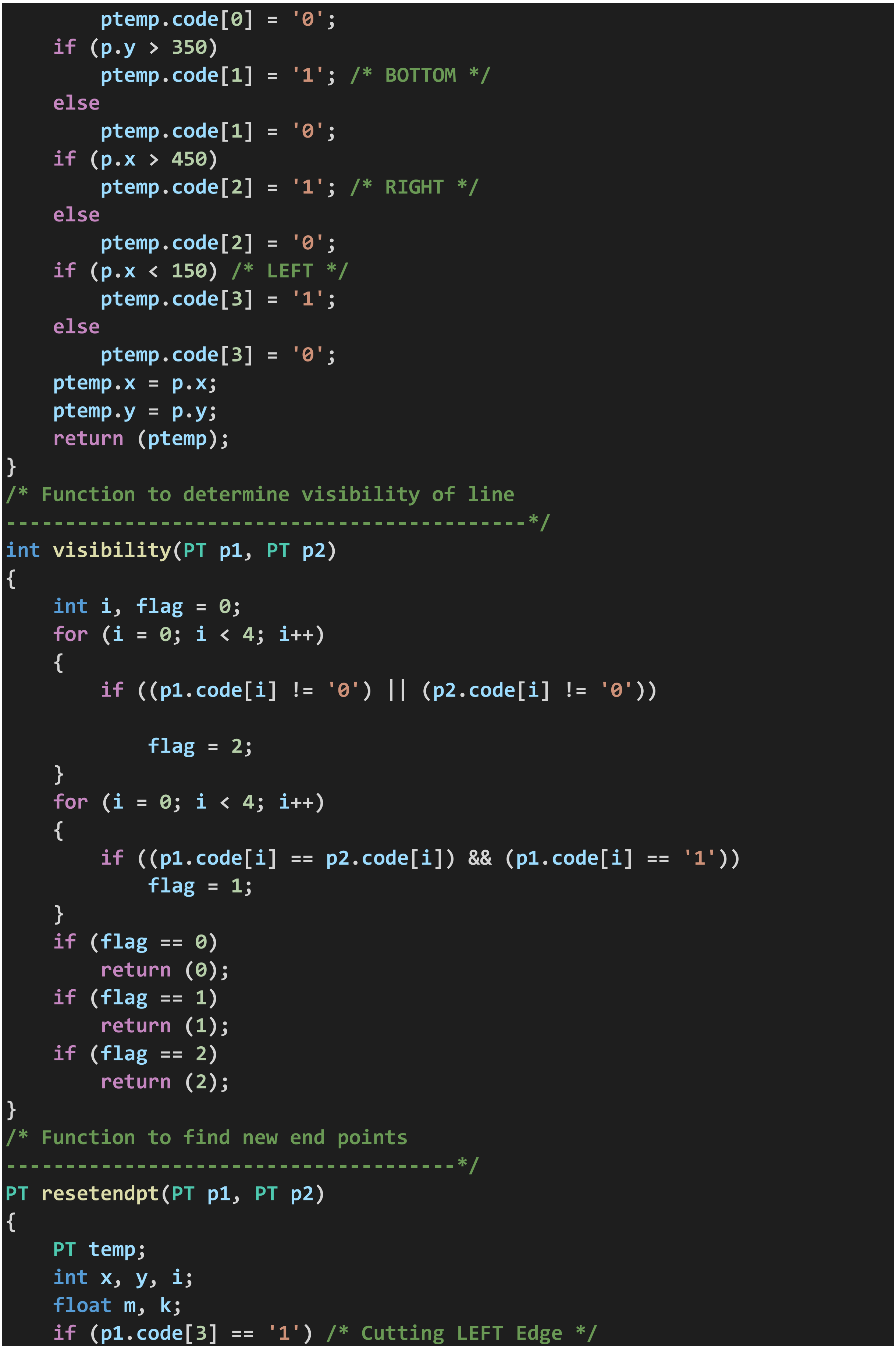
```
#include <iostream.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
/* Defining structure for end point of line */
typedef struct coordinate
{
    int x;
    int y;
    char code[4];
} PT;
void drawwindow();
void drawline(PT p1, PT p2, int cl);
PT setcode(PT p);
int visibility(PT p1, PT p2);
PT resetendpt(PT p1, PT p2);
void check_line(PT p1, PT p2);
int main()
{
    int gd = DETECT, gm;
    PT p1, p2;
    cout << "\n\t\tENTER END-POINT 1 (x,y): ";
    cin >> p1.x >> p1.y;
    cout << "\n\t\tENTER END-POINT 2 (x,y): ";
    cin >> p2.x >> p2.y;
    initgraph(&gd, &gm, "\\Turboc3\\bgi");
    drawwindow();
    drawline(p1, p2, 15);
    check_line(p1, p2);
    return (0);
}
void check_line(PT p1, PT p2)
{
    int v;

    p1 = setcode(p1);
```



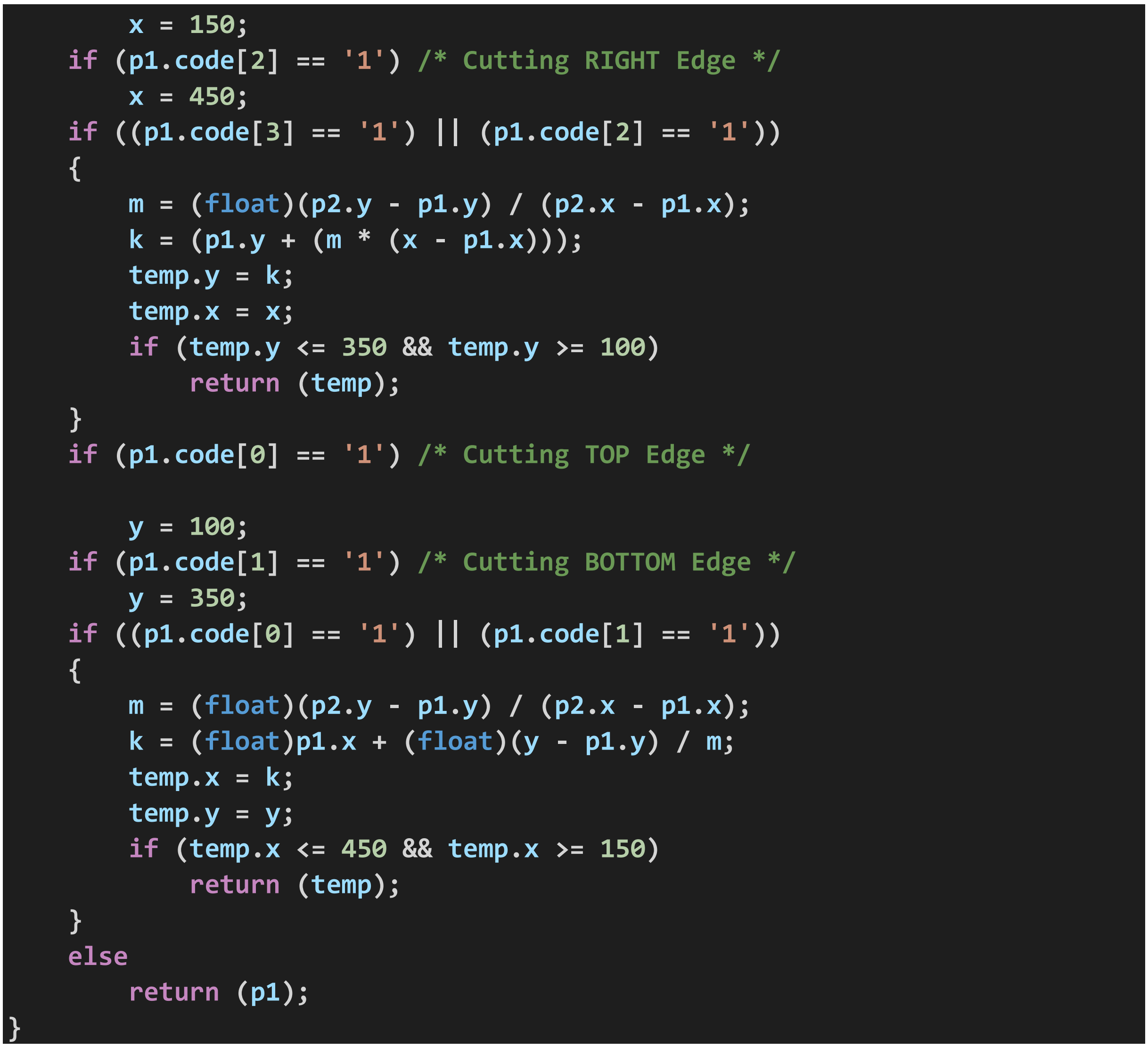
```c
    p2 = setcode(p2);
    v = visibility(p1, p2);
    switch (v)
    {
    case 0:
        cleardevice(); /* Line conpletely visible */
        drawwindow();
        drawline(p1, p2, 15);
        break;
    case 1:
        cleardevice(); /* Line completely invisible */
        drawwindow();
        break;
    case 2:
        cleardevice(); /* line partly visible */
        p1 = resetendpt(p1, p2);
        p2 = resetendpt(p2, p1);
        check_line(p1, p2);
        break;
    }
    delay(2000);
}
/* Function to draw window */
void drawwindow()
{
    setcolor(RED);
    line(150, 100, 450, 100);
    line(450, 100, 450, 350);
    line(450, 350, 150, 350);
    line(150, 350, 150, 100);
    delay(2000);
}
/* Function to draw line between two points
---------------------------------------------*/
void drawline(PT p1, PT p2, int cl)
{
    setcolor(cl);

    line(p1.x, p1.y, p2.x, p2.y);
    delay(2000);
} /* Function to set code of the coordinates
--------------------------------------------*/
PT setcode(PT p)
{
    PT ptemp;
    if (p.y < 100)
        ptemp.code[0] = '1'; /* TOP */
    else
```



```
        ptemp.code[0] = '0';
    if (p.y > 350)
        ptemp.code[1] = '1'; /* BOTTOM */
    else
        ptemp.code[1] = '0';
    if (p.x > 450)
        ptemp.code[2] = '1'; /* RIGHT */
    else
        ptemp.code[2] = '0';
    if (p.x < 150) /* LEFT */
        ptemp.code[3] = '1';
    else
        ptemp.code[3] = '0';
    ptemp.x = p.x;
    ptemp.y = p.y;
    return (ptemp);
}
/* Function to determine visibility of line
--------------------------------------------*/
int visibility(PT p1, PT p2)
{
    int i, flag = 0;
    for (i = 0; i < 4; i++)
    {
        if ((p1.code[i] != '0') || (p2.code[i] != '0'))

            flag = 2;
    }
    for (i = 0; i < 4; i++)
    {
        if ((p1.code[i] == p2.code[i]) && (p1.code[i] == '1'))
            flag = 1;
    }
    if (flag == 0)
        return (0);
    if (flag == 1)
        return (1);
    if (flag == 2)
        return (2);
}
/* Function to find new end points
--------------------------------------*/
PT resetendpt(PT p1, PT p2)
{
    PT temp;
    int x, y, i;
    float m, k;
    if (p1.code[3] == '1') /* Cutting LEFT Edge */
```



```
        x = 150;
    if (p1.code[2] == '1') /* Cutting RIGHT Edge */
        x = 450;
    if ((p1.code[3] == '1') || (p1.code[2] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (p1.y + (m * (x - p1.x)));
        temp.y = k;
        temp.x = x;
        if (temp.y <= 350 && temp.y >= 100)
            return (temp);
    }
    if (p1.code[0] == '1') /* Cutting TOP Edge */

        y = 100;
    if (p1.code[1] == '1') /* Cutting BOTTOM Edge */
        y = 350;
    if ((p1.code[0] == '1') || (p1.code[1] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (float)p1.x + (float)(y - p1.y) / m;
        temp.x = k;
        temp.y = y;
        if (temp.x <= 450 && temp.x >= 150)
            return (temp);
    }
    else
        return (p1);
}
```