ASSIGNMENT 6A

#Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
class demo{
public:
    void  plot(Ui::MainWindow*);
    void bresenham_circle(int xCenter, int yCenter, int radius,
Ui::MainWindow *);
    void dda_line(float x1, float y1, float x2, float y2, Ui::MainWindow *);
    void bresenham_line(int x1, int y1, int x2, int y2, Ui::MainWindow *);
    friend class MainWindow;
};

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    demo obj;

    void delay(int millisecondsTowait);

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

private:
    Ui::MainWindow *ui;
    friend class demo;
};
#endif // MAINWINDOW_H
```

#Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QColorDialog>
#include <QMessageBox>
#include <QTime>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
```

```
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
void delay(int millisecondsTowait){
    QTime dieTime=QTime::currentTime().addMSecs(millisecondsTowait);
    while(QTime::currentTime()<dieTime){
        QCoreApplication::processEvents(QEventLoop::AllEvents,100);
    }
}
QRgb c(qRgb(255,250,255));  //for white color(250 is for white and 0 is for black)
QImage k(500,500,QImage::Format_RGB888);

void demo::bresenham_circle(int xCenter, int yCenter, int radius, Ui::MainWindow *ui)
{
    int x = 0;
    int y = radius;
    int d = 3 - (2 * radius);  //intial decision parameter
    int x_arr[1000], y_arr[1000];
    int count = 0;
            while (x <= y)
            {
                if (d > 0)
                {
                    d = d + 4 * (x - y) + 10;
                    y--;
                }
                else
                {
                    d = d + (4 * x) + 6;
                }
                x++;
                k.setPixel(xCenter + x, yCenter - y, c); // this is the first octant
of the circle
                ui->label_4->setPixmap(QPixmap::fromImage(k));
                ui->label_4->show();
                x_arr[count] = x,
                y_arr[count] = y;
                count++;
                delay(20);
            }

//            for(int i = 0; i < count; i++)      //This is the second octant
//            {

//                k.setPixel(xCenter + y_arr[i], yCenter - x_arr[i], c);
//                delay(20);
//                ui->label_4->setPixmap(QPixmap::fromImage(k));
//                ui->label_4->show();
//            }
//            for(int i = 0; i < count; i++)      //These are the 3rd and 4th
quadrants. Reflection about x-axis.
//            {
//                k.setPixel(xCenter + x_arr[i], yCenter + y_arr[i], c);
//                k.setPixel(xCenter - x_arr[i], yCenter + y_arr[i], c);
//                k.setPixel(xCenter + y_arr[i], yCenter + x_arr[i], c);
//                k.setPixel(xCenter - y_arr[i], yCenter + x_arr[i],c);
//                delay(20);
//                ui->label_4->setPixmap(QPixmap::fromImage(k));
//                  ui->label_4->show();
//              }
//              for(int i = 0; i < count; i++)      //This is the second quadrant.
Reflection about y-axis.
//              {

//                  k.setPixel(xCenter - y_arr[i], yCenter - x_arr[i], c);
//                  k.setPixel(xCenter - x_arr[i], yCenter - y_arr[i], c);
//                  delay(20);
//                  ui->label_4->setPixmap(QPixmap::fromImage(k));
//                  ui->label_4->show();
//              }
            for(int i = 0; i < count; i++){
               k.setPixel(xCenter + y_arr[i], yCenter - x_arr[i], c);
               k.setPixel(xCenter + x_arr[i], yCenter + y_arr[i], c);
               k.setPixel(xCenter - x_arr[i], yCenter + y_arr[i], c);
               k.setPixel(xCenter + y_arr[i], yCenter + x_arr[i], c);
               k.setPixel(xCenter - y_arr[i], yCenter + x_arr[i],c);
               k.setPixel(xCenter - y_arr[i], yCenter - x_arr[i], c);
               k.setPixel(xCenter - x_arr[i], yCenter - y_arr[i], c);
               delay(20);
               ui->label_4->setPixmap(QPixmap::fromImage(k));
               ui->label_4->show();
            }
}

void demo::bresenham_line(int x1, int y1, int x2, int y2, Ui::MainWindow *ui)

{
    int dx = abs(x2 - x1);
        int sx = x1 < x2 ? 1 : -1; // condition to check if x1 > x2 then we need to
decrement
        int dy = -abs(y2 - y1);
        int sy = y1 < y2 ? 1 : -1;
        int err = dx + dy; // error value as we have taken dy negative so its dx - dy
        // err is to get the integer value of the nearest pixel

        while (true)
        {
            k.setPixel(x1, y1, c);
            if (x1 == x2 && y1 == y2)
                break;
            int e2 = 2 * err; // decision parameter
            if (e2 >= dy)
            {
                err += dy;
                x1 += sx;
            }
            if (e2 <= dx)
            {
                err += dx;
                y1 += sy;
            }
        }
        ui->label->setPixmap(QPixmap::fromImage(k));
}
void demo::dda_line(float x1, float y1, float x2, float y2, Ui::MainWindow *ui)
{
    float dx = (x2 - x1);
        float dy = (y2 - y1);
        float step = (abs(dx) >= abs(dy)) ? abs(dx) : abs(dy); // this is the no of
steps the program must run
        // also checks if slope > 1 or < 1.
        dx = dx / step; // this is the increment that must happen in each step
        dy = dy / step;

        float x = x1, y = y1;
        int i = 1;

        while (i <= step) // repeat the loop till step gets over
        {
            k.setPixel(x, y, c);
            x += dx;
            y += dy;
            i++;
        }

        ui->label->setPixmap(QPixmap::fromImage(k));
}//hi
void MainWindow::on_pushButton_clicked()
{
    QMessageBox message, msg, msg1;             //Exception handling to check the
entered values. Accepts integers only from 1 to 400.
        if(ui->textEdit->toPlainText().isEmpty()||ui->textEdit_2-
>toPlainText().isEmpty()||ui->textEdit_3->toPlainText().isEmpty())
        {
            message.information(0,"Error!","Empty Inbox");
        }
        else if(ui->textEdit->toPlainText().toInt() > 500 || ui->textEdit_2-
>toPlainText().toInt() > 500 || ui->textEdit_3->toPlainText().toInt() > 500 )
        {
            msg.information(0,"Error","Enter integer values between 1 to 400 only!");
        }
        else if(ui->textEdit->toPlainText().toUInt() && ui->textEdit_2-
>toPlainText().toUInt() && ui->textEdit_3->toPlainText().toUInt())
        {
            obj.plot(ui);
        }
        else
        {
            msg1.information(0,"Error","Enter integer values between 1 to 500!");
        }
}
void demo::plot(Ui::MainWindow *ui){
    int xC = ui->textEdit->toPlainText().toInt();
    int yC = ui->textEdit_2->toPlainText().toInt();
    int rad = ui->textEdit_3->toPlainText().toInt();

    float x1 = xC + rad * (float)sqrt(3) / 2;
    float x2 = xC - rad * (float)sqrt(3) / 2;
    float y1 = yC + rad / 2;

    bresenham_circle(xC, yC, rad, ui);       //for drawing outer circle
    bresenham_circle(xC, yC, rad / 2, ui);          //for drawing inner circle

    dda_line(x1, y1, x2, y1, ui);
    dda_line(x1, y1, xC, (float)yC - rad, ui);
    bresenham_line(x2, y1, (float)xC, (float)yC - rad, ui);          //left side of
the triangle

}
void MainWindow::on_pushButton_2_clicked()
{
    QRgb Color(QColorDialog::getColor().rgb());
    c=Color;
```

```
}


void MainWindow::on_pushButton_3_clicked()
{
    for (int x=0;x<500;++x){
        for (int y=0;y<500;++y){
            k.setPixel(x,y,qRgb(0,0,0));
        }
    }
    ui->label_4->setPixmap(QPixmap::fromImage(k));
}
```

ASSIGNMENT 6B

#Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void bresenham_circle(int x_center,int y_center,int radius);

    void DDA_Line(float x1,float y1,float x2,float y2);

    void bresenham_line(int x1,int y1,int y2,int x2);
    void assignment();

    void delay(int millisecondsTowait);

private slots:


    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();



private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

#Mainwindow.cpp

```
//Bresenham's circle drawing algorithm.

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QColorDialog>
#include <QMessageBox>
#include <QTime>

//Initialization of display screen
QImage img(600 , 600, QImage::Format_RGB888); //for white color(250 is for white and 0
is for black)
QRgb rgb(qRgb(255, 255, 255));

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
void delay(int millisecondsTowait){
    QTime dieTime=QTime::currentTime().addMSecs(millisecondsTowait);
    while(QTime::currentTime()<dieTime){
        QCoreApplication::processEvents(QEventLoop::AllEvents,100);
    }
}
void MainWindow::on_pushButton_clicked()
{
    assignment();
}

void MainWindow::assignment()
{
    // Taking input of diagonal coordinates from the user
    QMessageBox message, message2,message3,msg3;

    if (ui->textEdit->toPlainText().isEmpty() || ui->textEdit_2-
>toPlainText().isEmpty() || ui->textEdit_3->toPlainText().isEmpty() || ui->textEdit_4-
>toPlainText().isEmpty()) {
        message.information(0,"Error","Enter value in all the boxes");
    }

//    else if(ui->textEdit->toPlainText().isSimpleText() &&  ui->textEdit_2-
>toPlainText().isSimpleText()  &&  ui->textEdit_3->toPlainText().isSimpleText()){
//        message2.information(0,"Error","Enter Numerical values only");
//    }

    else if(ui->textEdit->toPlainText().toInt() > 600 || ui->textEdit_2-
>toPlainText().toInt() > 600 || ui->textEdit_3->toPlainText().toInt() > 600 || ui-
>textEdit_4->toPlainText().toInt() > 600){
        message3.information(0,"Error","The entered value is out of bounds");
    }
    else if(ui->textEdit->toPlainText().toInt() &&ui->textEdit_2-
>toPlainText().toInt() && ui->textEdit_3->toPlainText().toInt() && ui->textEdit_4-
>toPlainText().toInt())
```

```
    {
        int x1 = ui->textEdit->toPlainText().toInt();
        int y1 = ui->textEdit_2->toPlainText().toInt();
        int x2 = ui->textEdit_3->toPlainText().toInt();
        int y2 = ui->textEdit_4->toPlainText().toInt();

    // Calculating the center of the inside circle
        int xC = (x1 + x2)/2;
        int yC = (y1 + y2)/2;

    // Drawing the outside quadrilateral using DDA line drawing Algorithm function
        DDA_Line(x1,y1,x2,y1);
        DDA_Line(x1,y1,x1,y2);
        DDA_Line(x1,y2,x2,y2);
        DDA_Line(x2,y1,x2,y2);

    // Drawing the inside quadrilateral using DDA line drawing Algorithm function
        DDA_Line(xC,y1,x1,yC);
        DDA_Line(x1,yC,xC,y2);
        DDA_Line(xC,y2,x2,yC);
        DDA_Line(x2,yC,xC,y1);


    // Calculating radius of circle
        int xi = (x1-xC)/2;
        int yi = (y1-yC)/2;
        int xsq = xi*xi;
        int ysq = yi*yi;
        int rad = sqrt((xsq) + (ysq));

    //Drawing circle using Bressemham Circle Drawing Algorithm function
        bresenham_circle(xC,yC,rad);

    }
    else{
        msg3.information(0,"Error","Enter the Numberic Vaule!");  //other than intger
won't work
    }
}
// Implementation of Bressemham Circle Drawing Algorithm
void MainWindow::bresenham_circle(int xCenter, int yCenter, int radius)
{
    //in this method next pixel selected is that one who is at least distance
    int x = 0;
    int y = radius;
    int d = 3 - (2*radius); //Initial dicision parameter

    //Algorithm
    while (x <= y) {

        if(d > 0)
        {
            d = d + 4 * (x-y) + 10;//next decition paratmetr
            y--;
        }
        else {
            d = d + ( 4 * x) + 6;//next decition paratmetr
        }
        x++;
        img.setPixel(xCenter + x, yCenter + y , rgb);   //octet-1
        img.setPixel(xCenter - x, yCenter + y , rgb);   //octet-2
        img.setPixel(xCenter + x, yCenter - y , rgb);   //octet-3
        img.setPixel(xCenter - x, yCenter - y , rgb);   //octet-4
        img.setPixel(xCenter + y, yCenter + x , rgb);   //octet-5
        img.setPixel(xCenter + y, yCenter - x , rgb);   //octet-6
        img.setPixel(xCenter - y, yCenter + x , rgb);   //octet-7
        img.setPixel(xCenter - y, yCenter - x , rgb);   //octet-8
    }

    ui->label->setPixmap(QPixmap::fromImage(img));
    ui->label->show();
}

// Implementation of DDA line drawing Algorithm
void MainWindow::DDA_Line(float x1, float y1, float x2, float y2)
{
    float dx = (x2-x1);
    float dy = (y2-y1);   //   m(slope)=dy/dx
    float step = 0;
    if (abs(dx) >= abs(dy)){   //absoulute vaule
        step = abs(dx);
    }else {
        step = abs(dy);
    }

     dx=dx/step;
     dy=dy/step;
     float x=x1,y=y1;
     int i = 1;

     while (i <= step) {
         img.setPixel(x,y,rgb);
         x += dx;
         y += dy;
         i++;
     }
     ui->label->setPixmap(QPixmap::fromImage(img));
}


// Colour selector button
void MainWindow::on_pushButton_2_clicked()
{
    QRgb color (QColorDialog::getColor().rgb());
    rgb = color;
}


// Clear Screen buttton
void MainWindow::on_pushButton_3_clicked()
{
    for (int x = 0; x < 600; ++x)
    {
    for (int y = 0; y < 600; ++y)
     {
       img.setPixel(x, y, qRgb(0, 0, 0));
     }
   }
  ui->label->setPixmap(QPixmap::fromImage(img));
}
```

OUTPUT: