Family Name:.............................. **Other Names:** ....

ID Number: .... **Signature**.................................

CGRA 151: Terms Test

23rd August 2018

Instructions

- Time allowed: **45 minutes** .
- Answer **all** the questions. There are four questions and 45 marks in total.
- Write your answers in the boxes in this test paper and hand in pages 1–14.
- There is brief Processing documentation on page 15 which you may detach from the test paper.
- If you think some question is unclear, ask for clarification.
- This test contributes 10% of your final grade
- You may use paper translation dictionaries and approved calculators.
- You may write notes and working on this paper, but make sure your answers are clear.

Questions	**Marks**	
1. Graphical Output in Processing	[10]	
2. Straight Line Algorithm	[10]	
3. Vectors, Matrices and Transformations	[15]	
4. Bézier Cubic Curve	[10]	
	TOTAL	

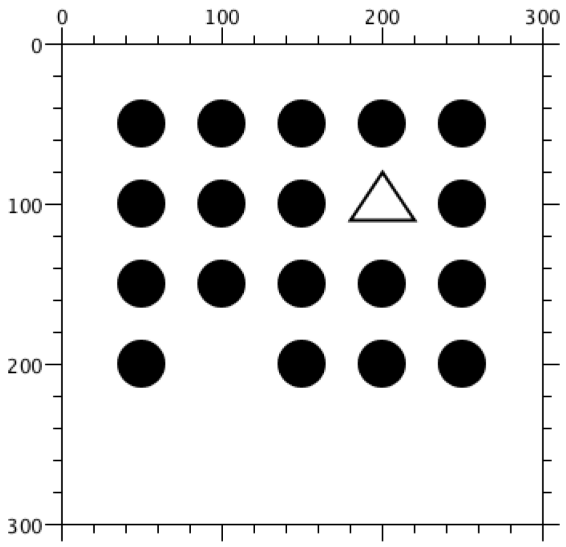
SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

1. **Graphical output in Processing** **(10 marks)**

   (a) **(5 marks)** Writing Processing Code

   Write Processing code to draw the following in the Processing window. For full credit you need to use loops appropriately.



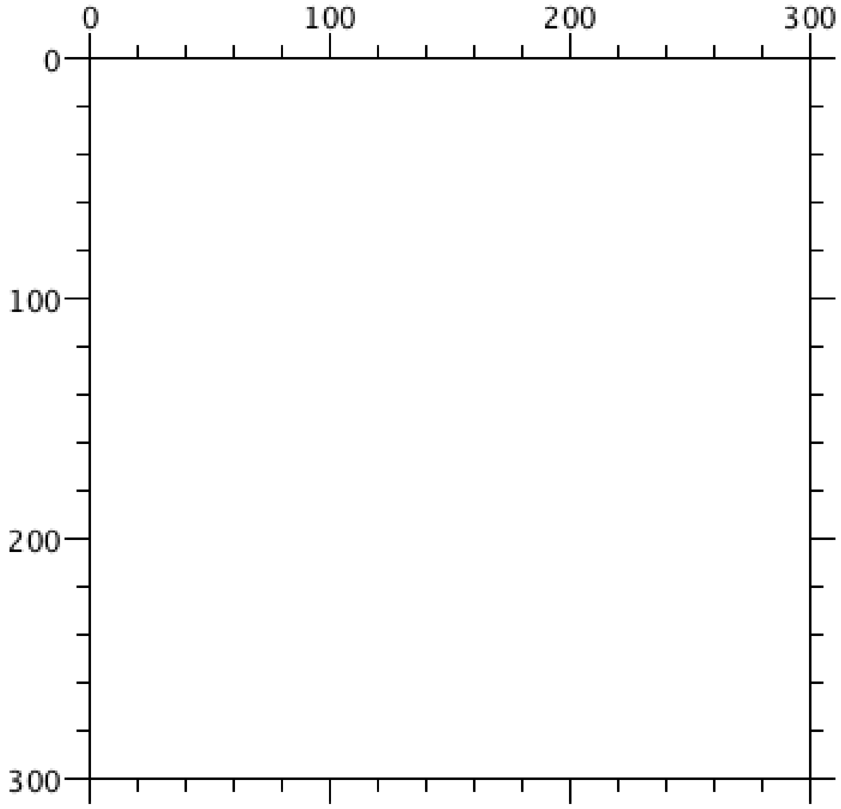
(the external grid and numbers are not part of the sketch, they are there for guidance)

```
size (300, 300);
















//
```

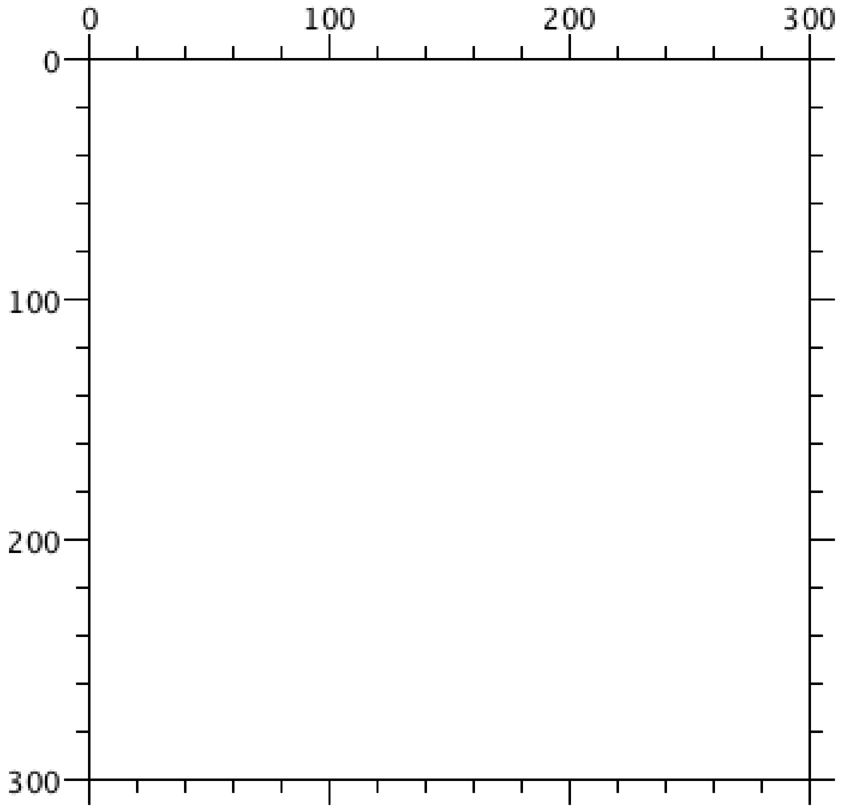
(Question 1 continued on next page)

(Question 1 continued)

Extra copies, provided in case you make a mistake in Question 1(b)



(spare copy of 300 × 300 Processing window)



(spare copy of 300 × 300 Processing window)

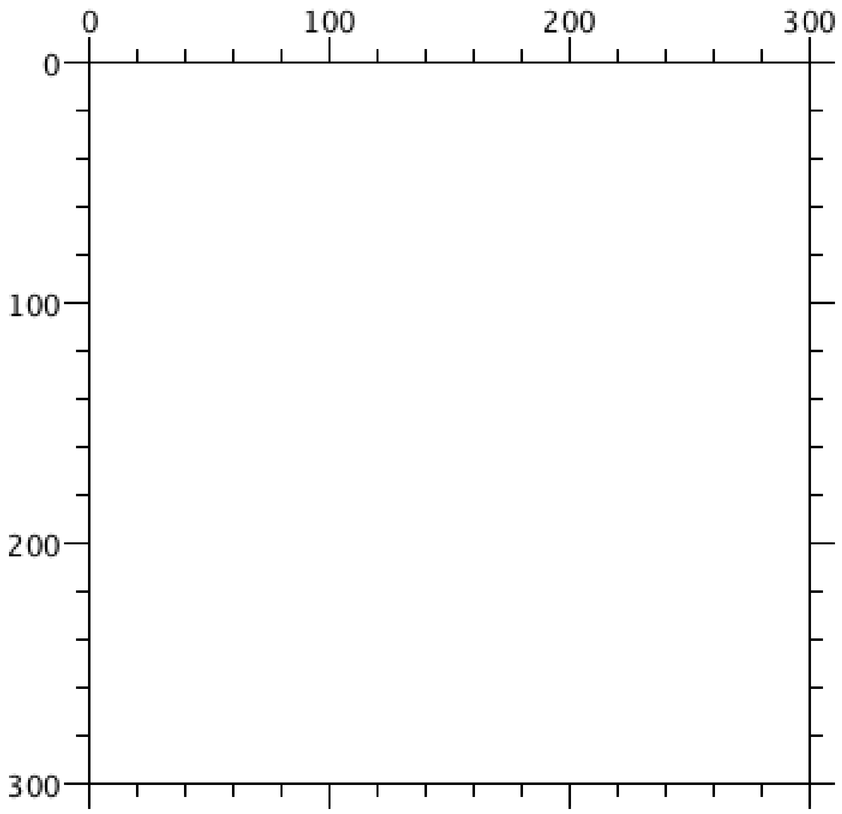
(Question 1 continued on next page)

(Question 1 continued)

(b) **(5 marks)** Interpreting Processing Code

Sketch what the following code will draw in the Processing window.

```
size( 300, 300 ) ;
background( 255 ) ;
fill(255);
stroke(0);
ellipse(100,80,60,60);
ellipse(100,160,100,100);
ellipse(100,240,120,120);
line(20,100,70,140);

noFill();
arc(130,100,120,80,0,HALF_PI);

fill(0);
noStroke();
ellipse(90,70,10,10);
ellipse(110,70,10,10);
triangle(100,80,90,100,110,90);
```



(300 × 300 Processing window)

2. **Straight line algorithm** **(10 marks)**

The following code draws a straight line by drawing the correct individual pixels.

Hint: note that this code draws one pixel in each *row* of pixels, unlike the examples in the lecture notes which have one pixel in each *column*.

```
void drawLine( int x0, int y0, int x1, int y1 ) {
        // get the inverse slope of the line , need 1.0 in there to prevent integer division
        float m = (1.0*(x1-x0))/(y1-y0);
        // set up variables for the integer and fractional parts of x
        float xi = 0.0 ;
        int x = x0 ;
        int y = y0 ;
        drawPixel(x,y);
        // ensure that the loop terminates after drawing the last pixel
        // not after the last-but-one or the one-after-the-last
        while (y < y1) {
              y++ ;
              xi += m ;
              if( xi > 0.5 ) {
                   xi -= 1.0 ; // the fractional part of x must be betwen -0.5 and 0.5
                   x += 1 ;
              }
              drawPixel(x,y) ;
        }
}
```

(a) **(3 marks)** Slope of the line.

What is the range of values for the slope of the line to be drawn by this algorithm?

(Question 2 continued on next page)

(Question 2 continued)

(b) **(7 marks)** Converting to using only integer arithmetic.

Improve this algorithm by avoiding using any floating point numbers (i.e., convert the algorithm from what is given to being Bresenham's line drawing algorithm in this octant, using only integer arithmetic).

```
void drawLine( int x0, int y0, int x1, int y1 ) {

}
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

3. **Vectors, Matrix and Transformation** **(15 marks)**

(a) **(7 marks)** Vector and matrix algebra

Given $\mathbf{p} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$, $\mathbf{q} = \begin{bmatrix} -8 \\ 10 \end{bmatrix}$, $\mathbf{r} = \begin{bmatrix} 11 \\ 3 \\ 1 \end{bmatrix}$, $\mathbf{M_1} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{M_2} = \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}$,

(i) (5 marks) Evaluate the following expressions.

Vector addition	$\mathbf{p} + \mathbf{q} =$	
Dot product	$\mathbf{p} \cdot \mathbf{q} =$	
Magnitude	$\lvert\mathbf{p} + \mathbf{q}\rvert =$	
Matrix multiplication	$\mathbf{M_1} \times \mathbf{M_2} =$	
Matrix multiplication	$\mathbf{M_2}\mathbf{r} =$	

(ii) (2 marks) What are the transformation commands in Processing that would produce the same result as the transformations represented by the matrices $\mathbf{M_1}$ and $\mathbf{M_2}$.

$\mathbf{M_1}$	
$\mathbf{M_2}$	

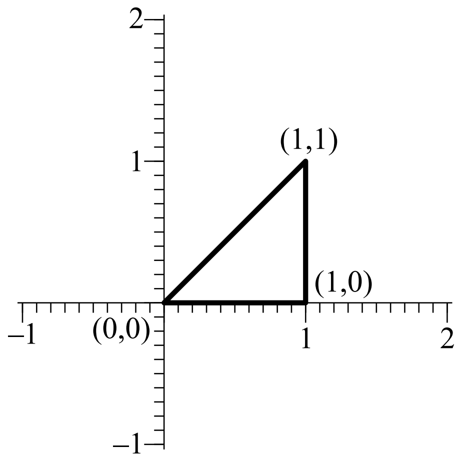
(Question 3 continued on next page)

(Question 3 continued)

(b) **(8 marks)** Transformation

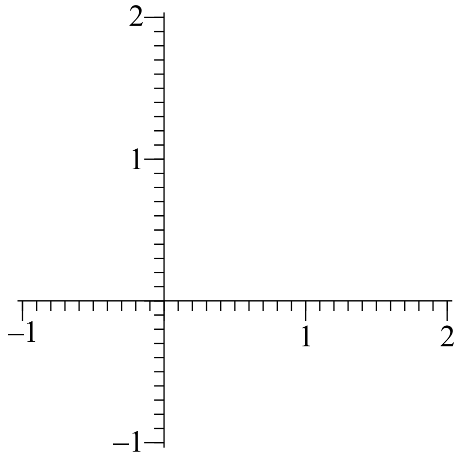
The diagram shows a triangle with a right angle. Apply the transformation represented by M_3 to this triangle. If necessary use the approximation $\frac{\sqrt{3}}{2} = 0.866$.

$$\mathbf{M_3} = \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} & 1 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



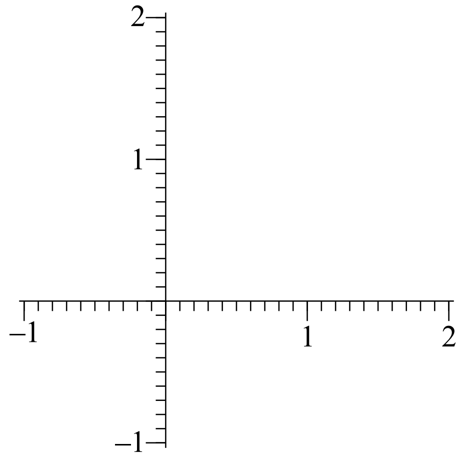
(i) (4 marks) Perform the calculations to transform each of the three vertices of the triangle.

(ii) (3 marks) Draw the transformed triangle. Label the three vertices of the transformed triangle with their transformed (x, y) coordinates.



(iii) (1 mark) What is the transformation represented by the matrix M_3.

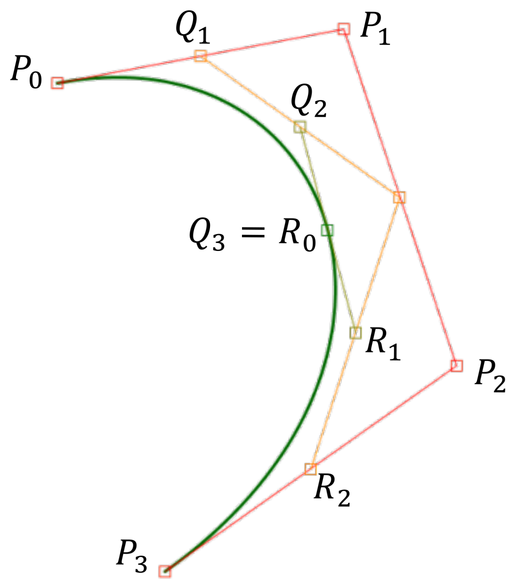
Extra copy, provided in case you make a mistake in Question 3(b)(ii)



(spare copy of coordinates for 3(b)(ii))

4. **Bézier Cubic Curve** **(10 marks)**

A Bézier cubic curve can be split into two smaller Bézier curves. When drawing a Bézier curve on the screen, we recursively do splitting until every piece can be approximated by a straight line. A Bézier curve is defined by four control points P_0, P_1, P_2, P_3. It can be split into two Bézier cubic curves defined by Q_0, Q_1, Q_2, Q_3 and R_0, R_1, R_2, R_3, where $Q_0 = P_0$ and $R_3 = P_3$. The point $Q_3 = R_0$ is the point where $t = \frac{1}{2}$ on the Bézier curve defined by P_0, P_1, P_2, P_3.

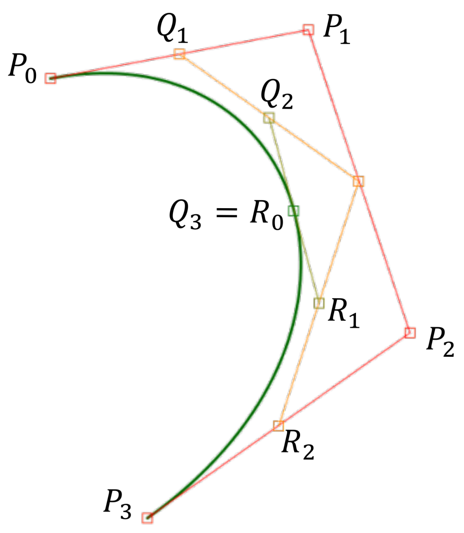


(a) **(5 marks)** Give formulas for the position of the control points Q_1, Q_2, Q_3, R_0, R_1, R_2 in terms of the points on the original curve P_0, P_1, P_2, P_3. For example, if you were asked to give a formula for R_3 you could write $R_3 = P_3$. Please write the representation of all six of the required points:

(Question 4 continued on next page)

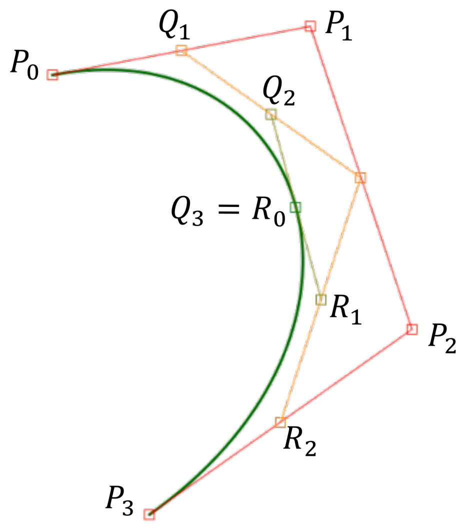
(Question 4 continued)

(b) **(2 marks)** On the diagram below draw the straight line that would be used to approximate the upper sub-curve (the curve defined by $Q_0 = P_0$, Q_1, Q_2, Q_3) [1 mark].
In order to determine whether the straight line approximation is to be used, we need to determine the distances from points Q_1 and Q_2 to that straight line. On the diagram below indicate the location of the point, C, on the straight line that we must find to allow us to determine the distance from Q_1 to the straight line. [1 mark]



(c) **(3 marks)** Give a formula that allows us to compute the position of the point C from the other points shown in the diagram.

Extra copy, provided in case you make a mistake in Question 4(b)



Brief Processing documentation

```
void point( float x, float y )
// draw a point at location (x,y)

void line( float x1, float y1, float x2, float y2)
// draw a line segment from location (x1,y1) to location (x2,y2)

void rect( float left, float top, float width, float height )
// draw a rectangle with its top left corner at the indicated position
// of size width times height

void ellipse( float centerX, float centerY, float width, float height )
// draw an ellipse centred at the indicated position
// of size width times height

void triangle( float x1, float y1, float x2, float y2, float x3, float y3 )
// draw a triangle with vertices at the three points
// (x1,y1), (x2,y2), (x3, y3)

void background( float grey )
void stroke( float grey )
void fill( float grey )
// set the colour of the background, stroke or fill to be
// a grey value between 0 (black) and 255 (white)

void noStroke()
void noFill()
// turn off drawing strokes around objects and filling objects , respectively

void rotate( float angle )
// rotate the universe by angle radians clockwise

void scale( float s )
// scale the universe by a factor of s
void scale( float sx, float sy )
// scale the universe by sx in the x-direction and sy in the y-direction

translate( float tx, float ty )
// translate the universe a distance (tx, ty)
```

* * * * * * * * * * * * * * *