Instructions:

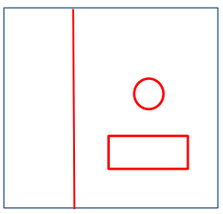
1. Answer all questions on this paper. For short answer questions, write, or circle, your answer in the space provided.
2. No aids of any kind (such as calculators or language translators or phones) are permitted.
3. You have 3 hours to complete the exam. There are a total of 72 marks.

Part 1: Short answer (32 marks)

[2] 1) What will be drawn in the canvas by the program below?

```
size(600,600);
background(255);
line(200,0,200, 600);
ellipse(400,300,100,100);
rect(300,400,200,100);
```

Canvas:



[2] 2) What will be printed in the console by the following program?

```
float x = 9;
int y = 2;
println(x / y);
println(y % 3);
```

Console:

4.5
2

[2] 3) Study the code below. Then list all the possible widths of the rectangles.

```
int x = 1;
void draw () {
   x = (x % 3) + 10;
   rect(mouseX, mouseY, x, x);
}
```

10,11,12

[2] 4) In the code below, draw an arrow from each use of a variable (the four places are shown in **bold**) to the declaration of that variable. One arrow is shown for you.

```
int x = 3;
float y = 2;

void setup(){
   float y=2;
   fn(x + y);
}
void fn(float x){
   y = 0;
   float y = 5 + x;
   println(y);
}
```

[2] 5) Write a single IF statement that will match the comment below.

```
//Reset count to 0 if x is a multiple of 10, or y is a multiple of 10,
//or both x and y are multiples of 5.

if(x%10==0 || y%10==0 || (x%5==0 && y%5==0))
   count = 0;
```

[2] 6) What will be printed in the console by the following program?

```
float x = 3.2;
float y = 2.3;
boolean a = x <= y;
boolean b = 2*y > x;
if(a || !b) {
   if(b && !a)
      println("one");
   else
      println("two");
}
else
   if(b && !a)
      println("three");
   else
      println("four");
```

Console:

three

[2] 7) When should you use a do-while loop instead of a while loop?

_ When the statement within the loop MUST be done AT LEAST ONCE_____

[2] 8) Complete the code to print a random number between 1 and 100 whose last digit is a 5 or an 8.

```
int n;
int lastDigit;
do {
   n = __int(random(1,101));_____________________________
   lastDigit = n % 10;
} while(_____lastDigit != 5 && lastDigit != 8_____________________);
println(n + " ends with " + lastDigit);
```

[2] 9) There is an **int** variable **score** in a program. Give the statements that will draw the value of **score** in the centre of the canvas.

- Use text 24 pixels tall.
- Draw it in red.
- You can use **height/2** for the **y** coordinate, but adjust the **x** coordinate so that the number is horizontally centred in the canvas.

```
textSize(24);
fill(255,0,0);
String s = str(score);
text(s,(width-textWidth(s))/2,height/2);
```

[2] 10) Complete the code to print the average word length of the words in the String **words**. (For the example below, the average would be 2.8.) The String will only contain letters and commas, with no comma after the last word. It must work for *any* such String. Do not use a loop.

```
String words = "the,cat,in,the,hat";
int numCommas = 4;

println ("The average word length is: " +

_________ (float)(words.length()-numCommas)/(numCommas+1)_____________________);
```

[2] 11) What will be printed in the console by the following program?

```
float w = 2.2;
void setup(){
   float y = 1.0;
   float x = fn(w + y, y);
   println(x + "," + y + "," + w);
}
float fn(float x, float y){
   w = 0.5;
   y = 2.0;
   return w + x + y;
}
```

Console:

5.7,1.0,0.5

[1] 12) Why are functions helpful? Circle your answer.

a) Functions make code more readable.
b) Functions allow for code re-use.
c) Functions break a large problem into manageable chunks.
d) Functions reduce the size of the setup and draw functions.
e) Functions are helpful for all of the above reasons.

[1] 13) Given the following function header, which of the following function calls would be legal?

```
String makeStr(char fillChar, int size) { …
```

a)
```
double d = makeStr('A', 10);
```
b)
```
println(makeStr(' ', 50));
```
c)
```
String s = makeStr("FILL!", 10);
```
d)
```
char c = makeStr(char fillChar, int size);
```
e)
```
String s = makeStr('a', 3.14159);
```

[2] 14) What will be printed in the console by the following program?

```
int[] a = {1,2,3};
void setup(){
   test1(a);
   println(a[0] + "," + a[1]);
   test2(a);
   println(a[0] + "," + a[1]);
}
void test1(int[] x){
   x = new int[5];
}
void test2(int[] x){
   x[0] = 5;
}
```

Console:

1,2
5,2

[1] 15) What does the array **arr** contain after running the following code?

```
int[] arr = {1, 2, 3, 4, 5, 6, 7};
for (int i = 0; i < arr.length / 2; i++) {
   int temp = arr[arr.length - 1 - i];
   arr[arr.length - 1 - i] = arr[i];
   arr[i] = temp;
}
```

a) `{1, 2, 3, 4, 3, 2, 1}`
b) `{1, 2, 3, 4, 5, 6, 7}`
c) `{7, 6, 5, 4, 3, 2, 1}`
d) `{1, 1, 1, 1, 1, 1, 1}`
e) `{1, 3, 5, 7, 2, 4, 6}`

[1] 16) Given the following arrays, which statement will make **y** a deep copy of **x**?

```
int[] x = {1, 2, 3};
int[] y = new int[x.length];
```

a) `y = x;`
b) `y[] = x[];`
c) `y = x.copyArray();`
d) `for (int i = 0; i < x.length; i++) { y = x; }`
e) `for (int i = 0; i < x.length; i++) { y[i] = x[i]; }`

[2] 17) Give the main advantage and the main disadvantage of using a binary search instead of a linear search.

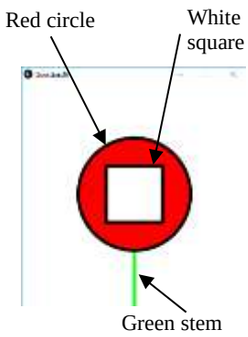
Advantage: _____**Much faster**_____

Disadvantage: _____**Requires a sorted list**_____

[Questions 18 and 19 have been deleted.]

Part 2: Programming (40 marks)

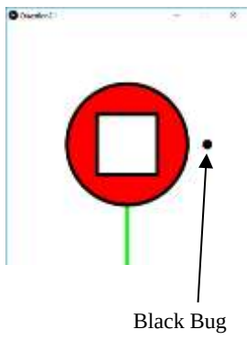
[5] 20) It's spring! Complete the code below to draw a flower.



- Use a 400x400 canvas with a white background, and thick lines (6 pixels wide). [1 mark]
- Set the colours as indicated (red, white, black, green) [1 mark]
- The stem should be a green line from the centre of the circle to the bottom of the canvas [1 mark]
- Draw a red circle with a black border for the flower. Its centre should be in the centre of the canvas. Its diameter should be ½ the width of the canvas. [1 mark]
- Draw a white square with a black border in the middle of the red circle, with a width that is half the diameter of the circle. [1 mark]

```
size(400, 400);
background(255); //white background
strokeWeight(6); //thick lines, as shown
stroke(0,255,0);
line(width/2,height/2,width/2,height);
fill(255,0,0);
stroke(0);
ellipse(width/2,height/2,width/2,width/2);
fill(255);
rect(3*width/8,3*height/8,width/4,height/4);
```

[5] 21) With spring come bugs! Write a program to draw the flower from Question 20, with a bug that moves around the flower.



- The bug is a black circle that should rotate around the flower somewhere in the space between the edge of the red flower and the edge of the canvas. The bug should not touch the flower or the edge of the canvas. The exact position doesn't matter.
- The bug should start in the position shown, to the right of the flower.
- The bug should circle clockwise around the flower, taking 2 seconds to go once around the flower.
- The width and height of the bug should be 1/40th of the width of the canvas.
- You can assume that your code from Question 20 is in the place indicated, except for the **`size`** command. Put that one in its proper place.

```
float angle = 0;
final float SPEED = 2.0*PI/120;

void setup(){
  size(400, 400);
}

void draw(){



// ASSUME YOUR CODE TO DRAW THE FLOWER IS HERE
// YOU DO NOT NEED TO WRITE IT AGAIN




  fill(0);
  ellipse(width/2+width/3*cos(angle),
          height/2+width/3*sin(angle),
          width/40,width/40);
  angle += SPEED;

}
```

[5] 22) Complete the function below which will calculate and return the discount to be given to a customer, as a value from 0.0 (full price) to 1.0 (free). For example, if the customer should get a 20% discount, this function should return 0.20. The discount is based on the quantity of items ordered, and whether or not the customer is a "preferred customer". The rules are given below, and suitable constants have been defined for you.

- Preferred customers get
  - a 25% discount for 100 items or more
  - a 10% discount for fewer items.
- Regular customers get
  - a 15% discount for 200 items or more
  - a 5% discount for 50-199 items
  - no discount at all otherwise.

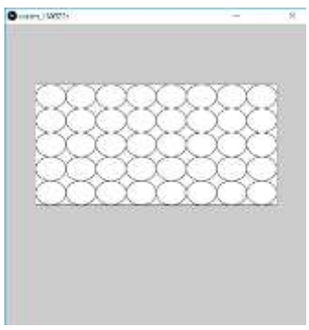
```
float discount(boolean preferred, int quantity){
   final float PREF_BULK = 0.25;
   final float PREF = 0.10;
   final float REG_BULK = 0.15;
   final float REG_LARGE = 0.05;

   float answer = 0.0;

   if(preferred)
      if(quantity>=100)
         answer = PREF_BULK;
      else
         answer = PREF;
   else
      if(quantity>=200)
         answer = REG_BULK;
      else if(quantity>=50)
         answer = REG_LARGE;
      else
         answer = 0.0; //not really needed

   return answer;
}
```

[5] 23) Complete the function below, which will draw a rectangle with the given top left corner, width, and height, and then fill it with the given number of rows and columns of ellipses. For example, the function call **`texturedRect(50,100,400,200,5,8);`** should produce the result shown at right, if the canvas is 500x500.



```
void texturedRect(int left, int top,
                  int wide, int high,
                  int rows, int cols){
   stroke(0);
   strokeWeight(1);
   fill(255);

   rect(left,top,wide,high);
   float ellipseWide = (float)wide/cols;
   float ellipseHigh = (float)high/rows;
   for(int r=0; r<rows; r++){
      float y = top+(r+0.5)*ellipseHigh;
      for(int c=0; c<cols; c++){
         float x = left+(c+0.5)*ellipseWide;
         ellipse(x,y,ellipseWide,ellipseHigh);
      }
   }
}
```

[5] 24) Complete the function below, which will accept a **`String s`**. If **`s`** contains a **`<`** character, and then a **`>`** character to the right of that, then this function should return a **`String`** containing all of the characters that appear between **`<`** and **`>`**. If there is no **`<`**, or no **`>`**, or if **`>`** appears before **`<`** instead of after it, then the function should return an empty **`String`**. You may assume that there are no more than one of each of **`<`** and **`>`** characters. For example:

`grabTag("Test <one> two")` should return **`"one"`**
`grabTag("Test <one two")` should return **`""`**
`grabTag("Test >one< two")` should return **`""`**

In this question you are only allowed to use the basic **`length`**, **`charAt`**, and **`+`** operations on **`String`**s. You can not use **`indexOf`**, **`substring`**, or any other **`String`** operations, even if you happen to know them.

```
String grabTag(String s){

   //There are quite a few ways that this could be done

   String answer = "";
   int open=-1, close=-1;
   for(int i=0; i<s.length(); i++)
      if(s.charAt(i)=='<')
         open = i;
      else if(s.charAt(i)=='>')
         close = i;
   if(open>=0 && close>open)
      for(int i=open+1; i<close; i++)
         answer +=s.charAt(i);
   return answer;
}
```

[5] 25) a) Write a function **hasComment** which will accept a **String** containing a single line of code from a Processing program. It should determine whether or not the line contains a **//** comment. If it does, it should return an **int** giving the position of the comment (the index of the first of the two **/** consecutive characters), otherwise it should return -1. For example,

hasComment("int noComment=0;") should return **-1**
hasComment("int x=width/2; //This is x") should return **15**

```
int hasComment(String line){
   for(int i=0; i<line.length()-1; i++) //important to stop one character early
      if(line.charAt(i)=='/' && line.charAt(i+1)=='/')
         return i;
   return -1;
}
```

b) Write a function **chopComment** which will also accept a **String** containing a single line of code from a Processing program. It should remove any **//** comment from the line, if there is one, returning a **String** containing the possibly modified line. It should, of course, use the **hasComment** function. For example,

chopComment("int noComment=0;") should return **"int noComment=0;"**
chopComment("int x=0; //This is x") should return **" int x=0; "**

```
String chopComment(String line){
   String newLine = "";
   int place = hasComment(line);
   if(place == -1)
      return line;
   else
      for(int i=0; i<place; i++)
         newLine += line.charAt(i);
   return newLine;
}
```

[5] 26) Mirror writing: Complete the program below that, whenever the mouse button is pressed, will store the path that the mouse is taking by recording the sequence of **`mouseX`** and **`mouseY`** values in arrays, *but will not draw anything.* When the mouse button is released the program should immediately draw the *mirror imag*e of whatever the user drew (flipped left-to-right). It should repeat the process every time the mouse button is pressed.

```
final int MAX_POINTS = 100000; //Allow a lot of points
int[] xCoords = new int[MAX_POINTS];
int[] yCoords = new int[MAX_POINTS];
int numPoints = 0;

void setup(){
   size(500,500);
}

void draw(){


   if(mousePressed) {
     //just collect the current mouseX,mouseY coordinates
     xCoords[numPoints] = mouseX;
     yCoords[numPoints] = mouseY;
     numPoints++;
   }
   else {
     //Draw the line stored in the arrays, and clear it out.
     for(int i=1; i<numPoints; i++)
        line(width-xCoords[i-1],yCoords[i-1],
             width-xCoords[i],yCoords[i]);
     numPoints = 0; //Start all over again next time.
   }









}//close draw
```

[5] 27) Complete the function **myBestSearch** below which should search **myArray** for **key**. If **key** is found, the function should return the index (position) where it occurs in **myArray**. If the **key** is not found in the array the function should return -1.
Note: The maximum mark is 5 if you do a binary search, and 3 if you do a linear search.

```
int myBestSearch (int[] myArray, int key){

  int lo=0;
  int hi=myArray.length-1;
  int mid;
  while(lo<=hi){
    mid=(lo+hi)/2;
    if(myArray[mid]==key)
      return mid;
    else if(myArray[mid]<key)
      lo=mid+1;
    else
      hi=mid-1;
  }//while
  return -1;
}//myBestSearch
```