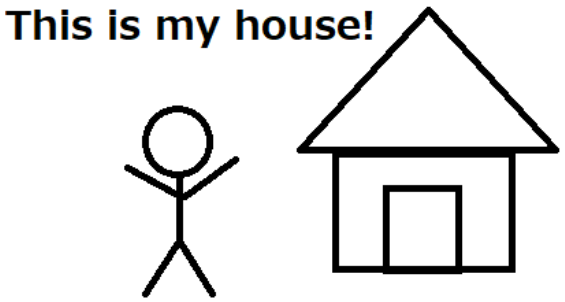
□ ～ Processing問題 ～

I. 図形の練習

1. 下記の絵を描きましょう。

※図形はtriangle, rect, ellipse, lineの各関数を使うこと。言葉はtextを使うこと。
文字サイズは textSize(32); などと変更できます。



II. 分岐処理(**if**文)の練習

1. キーを押すと色が変わる円

①size(600,400)の画面の中央に白い円を表示
②rキーを押すと赤色に変わる
③gキーを押すと緑色に変わる
④bキーを押すと青色に変わる

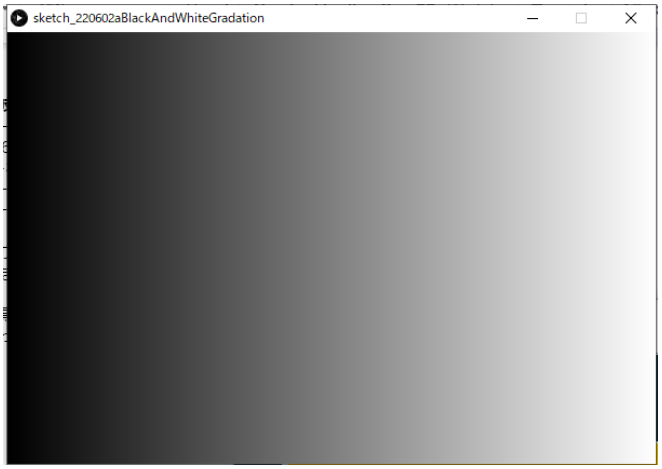
2. マウスを追いかける円

円を描き、マウスの後を追いかけるプログラムをつくりましょう。
現在の円の中心座標(xとy)とマウスの座標(mouseXとmouseY)を比較する。

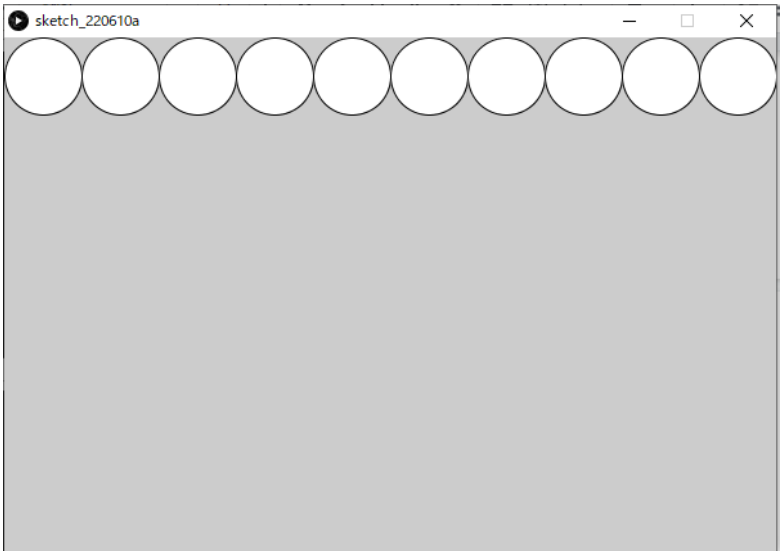
III. 繰り返し処理(**for**文)の練習

1. 白黒のグラデーション

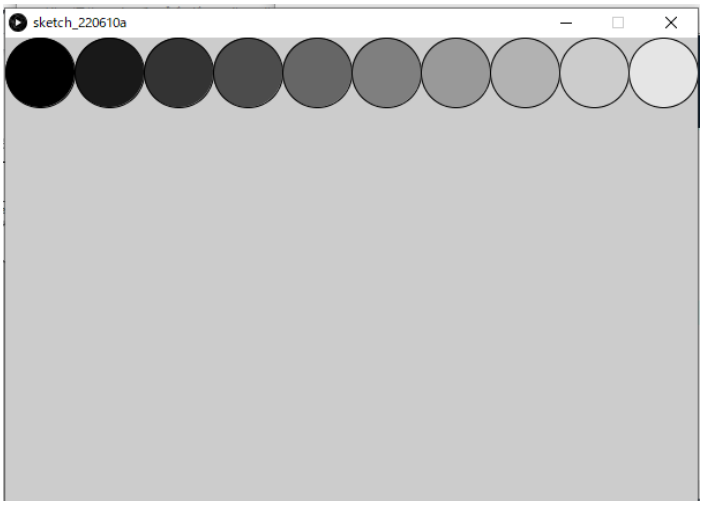
画面いっぱいに黒から白へ変化するグラデーションを描こう。
ヒント:line関数で縦線を描く。線の色はstroke関数で決める。
stroke関数の値を少しずつ変える。
たくさんの線はfor文で繰り返す。iの値は0からwidthまで。



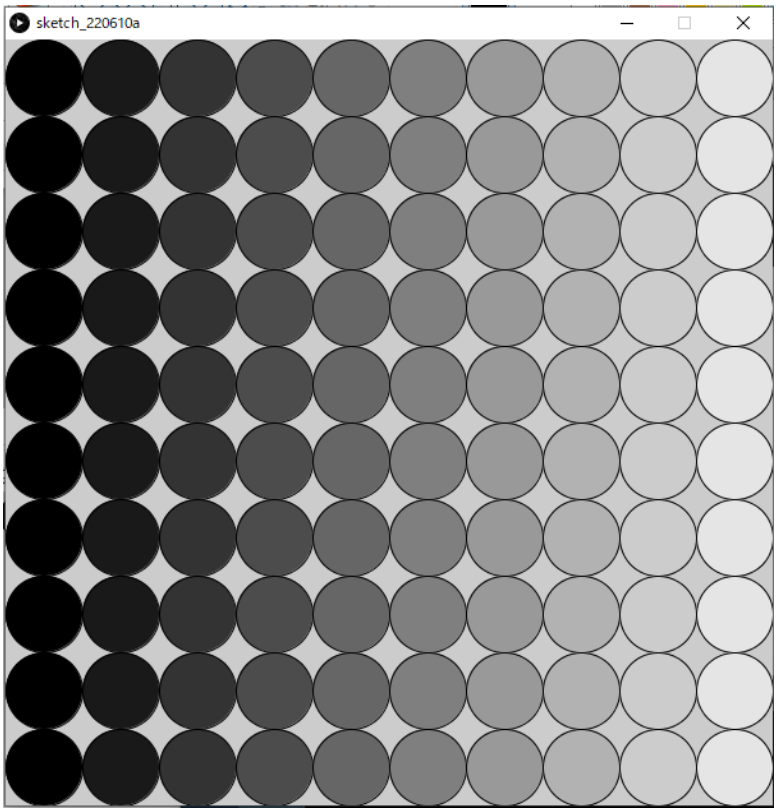
2. 横一列に10個の〇を描く。
size(600,400);
円の半径は30(直径は60)



3. グラデーションの色をつけてみる



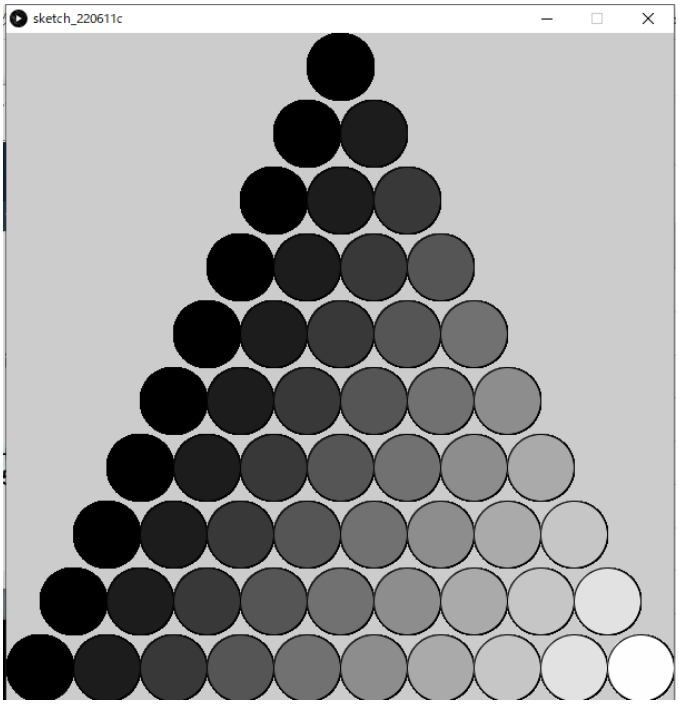
4. 縦方向にも繰り返す



5. ピラミッドをつくってみよう。10段。

ヒント：上からではなく下の行から描いていく。i++ではなくi--を使う。

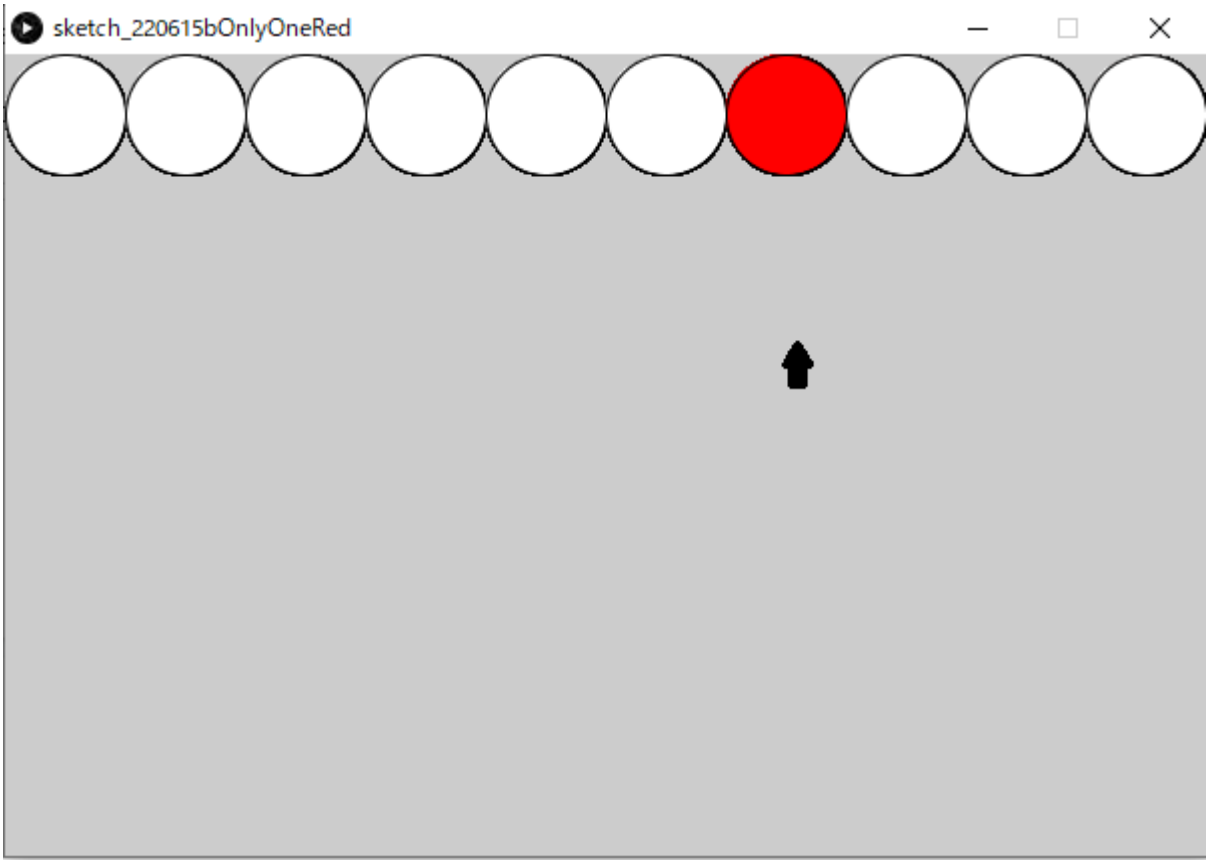
各段の最初の円の座標、隣同士の円の中心の差について考えましょう。



IV. 配列の中から最小値、最大値、平均値を求める
※下記プログラムを書いて実行してください。
日本語のコメントは書かなくて大丈夫です。

```
int[] ary = {29,5,14,22,8,32,11,3,18,17,21}; //整数11個の配列を作成

//この配列から最小値、最大値、平均値を求めます

//***** 最小値を見つける *****
int karinoMin = Integer.MAX_VALUE; //整数の最大値を仮の最小値にする(最初に必ず入れ替えるため)

for(int i=0; i<ary.length; i++){ // 配列変数名.lengthで配列の要素数を得られる(この場合は11)
  // 現在の要素の値が仮の最小値よりも小さければ仮の最小値をその値にする
  if(ary[i] < karinoMin){
    karinoMin = ary[i];
  }
}
//最終的に仮の最小値には本当の最小値が入っている
println("最小値:",karinoMin);

//***** 最大値を見つける *****
int karinoMax = Integer.MIN_VALUE; //整数の最小値を仮の最大値にする(最初に必ず入れ替えるため)

for(int i=0; i<ary.length; i++){
// 現在の要素の値が仮の最大値よりも大きければ仮の最大値をその値にする
  if(ary[i] > karinoMax){
    karinoMax = ary[i];
  }
}
//最終的に仮の最大値には本当の最大値が入っている
println("最大値:",karinoMax);

//***** 平均値を算出 *****
float ave = 0.0;
for(int i=0; i<ary.length; i++){
  ave = ave + ary[i]; //各要素の値を足しこんでいく
}
//変数aveには合計値が入っているので、要素数で割って平均値を求める
ave = ave / ary.length; //平均値を算出

//**** 小数第1位まで求めたい場合 ****
// round関数は小数点以下を四捨五入するため、小数第1位までを
// 求めたい場合は一度10倍してから丸めて、10.0で割り算する。
ave = round(ave * 10) / 10.0; //小数点以下を四捨五入

println("平均値:",ave);
```

1. for文を使って白い〇を横に10個並べて、マウスをクリックし、マウスがクリックされた場所に一番近い〇だけ赤い色になるようなプログラムを作成して下さい。
   ヒント　一番近い〇はマウスの位置と各〇の中心座標の距離が最小となるもの
   2点(x1,y1)と(x2,y2)の距離を求める関数　**dist(x1,y1,x2,y2)**　を使います。
   どの〇が一番近いかを調べるのは**mousePressed**関数で行います。



V. ソート(並べ替え)

1. バブルソート

ソートとは

ソートとは、「並べ替え」の意味です。数値データを、大きい順、もしくは小さい順に並べ替える処理のことを言います。

バブルソートはソートを行う方法の1つで、最も基本的なソートの方法です。他にシェーカーソート、クイックソートなどがあります。

なお、データを小さい順に並べ替えることを、昇順（しょうじゅん）、大きい順に並べ替えることを、降順(こうじゅん)と言います。

参考サイト

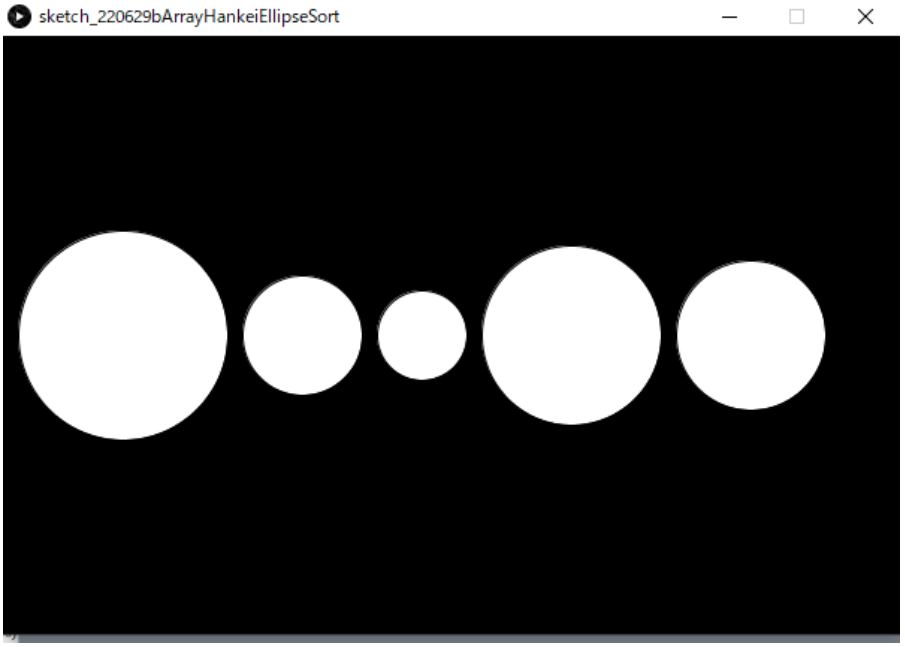
http://sevendays-study.com/algorithm/day5.html

```
int[] x = {8,2,7,1,10,6,5,3,4,2}; //整数型配列の定義
int tmp = 0;
for(int i=0;i<x.length-1;i++){
  for(int j=i+1;j<x.length;j++){
    if(x[i] > x[j]){ //昇順、降順は<とする。
      tmp = x[i]; //x[i]とx[j]を入れ替えるため一旦退避
      x[i] = x[j];
      x[j] = tmp;
    }
  }
  //配列の内容を確認
  for(int k=0; k<x.length; k++){
    print(x[k]);
    if(k < x.length - 1){
      print(",");
    }
  }
  println();
}
```

2. 問題
次のプログラムは異なる直径の円を5つ並べます。

```
int[] d = {70,40,30,60,50};
int t = 10;
int x = 0;
int y = 200;

void setup(){
  size(600,400);
}

void draw(){
  x = 0;
  background(0);
  for(int i=0; i<d.length; i++){
    x = x + d[i] + t;
    ellipse(x,y,d[i] * 2,d[i] * 2);
    x = x + d[i];
  }
}
```

【実行結果】



このプログラムを元に**a**キーを押されたら小さい半径順に、**d**キーを押されたら大きい半径順に
左から並べるプログラムを作成してください。

解答例
II
①.

```
float r;
float g;
float b;

void setup(){
  size(600,400);
  r = 255;
  g = 255;
  b = 255;
  }

void draw(){
  background(0);
  fill(r,g,b);
  ellipse(300,200, 100,100);
}

void keyPressed(){
  println(keyCode);
  if(keyCode == 82){
    r = 255;
    g = b = 0;
  }
  if(keyCode == 71){
    g = 255;
    r = b = 0;
  }
  if(keyCode == 66){
    b = 255;
    g = r = 0;
  }
}
```

②

```
int x = 0;
int y = 0;
int d = 50;
int speed = 3;
void setup(){
  size(600,400);
}

void draw(){
  background(0);
  if(x < mouseX){
```

```
    x +=speed;
  } else if(x > mouseX){
    x -=speed;
  }
  if(y < mouseY){
    y +=speed;
  } else if(y > mouseY){
    y -=speed;
  }

  fill(255);
  // dist関数  2点の距離を求める。
  // マウスポインタと円の中心の距離が半径以下のときに青色にする
  if(dist(x,y,mouseX,mouseY) <= d / 2){
     fill(0,0,255);
  }
  ellipse(x,y,d,d);
}
```

|||

①

```
int r=30;
int d=r*2;
size(600,400);
for(int i=0; i<width/d; i++){
  ellipse(r+d*i,r,d,d);;
}
```

②

```
int r=30;
int d=r*2;
size(600,400);
for(int i=0; i<width/d; i++){
  ellipse(r+d*i,r,d,d);;
}
```

③
```
int r=30;
int d=r*2;
float c = 0;
size(600,400);
for(int i=0; i<width/d; i++){
  fill(c);
  ellipse(r+d*i,r,d,d);;
  c = c + 255.0 / (width / d);
}
```

④
```
int r=30;
int d=r*2;
float c = 0;
size(600,600);
for(int i=0; i<height/d; i++){
  for(int j=0; j<width/d; j++){
    fill(c);
    ellipse(r+d*j,r+d*i,d,d);;
    c = c + 255.0 / (width / d);
  }
  c = 0;
}
```

別のプログラム
```
size(600,600);

for(int i=0;i<10;i++){
 float c = 0;
 for(int j=0;j<10;j++){
   c = j * 255.0/9;
   println("c=",c);
   fill(c);
   ellipse(30+j*60,30+i*60,60,60);
 }
}
```

⑤ピラミッド
```
size(600,600);

for(int i=10; i>0; i--){
  float c = 0;
  for(int j=0; j<i;j++){
    fill(c);
    int x = 30 + j * 60 + (10 - i) * 30;
```

```
    int y = i * 60 - 30;
    ellipse(x, y, 60, 60);
    c = c + 255.0 / 9;
  }
}
```

別のプログラム

```
for(int i=9;i>=0;i--){
 float c = 0;
 for(int j=0;j<10-(9-i);j++){
   int x = 30 * (10 - i) + j * 60;
   int y = 30 + i * 60;
   c = j * 255.0/9;
   println("c=",c);
   fill(c);
   ellipse(x,y,60,60);
 }
}
```

別のプログラム２

```
size(600,600);

int k = 0;
for(int i=9;i>=0;i--){
 float c = 0;
 for(int j=0;j<10-k;j++){
   int x = 30 * (10 - i) + j * 60;
   int y = 30 + i * 60;
   c = j * 255.0/9;
   println("c=",c);
   fill(c);
   ellipse(x,y,60,60);
 }
 k = k + 1;
}
```

IV
1
//マウスに近い〇のみ赤色にする

```
int r=30;
int d=r*2;
int nearest = -1;

void setup(){
  size(600,400);
}

void draw(){
  //白い〇を10個並べる
  for(int i=0; i<width/d; i++){
    if(i == nearest){
      //nearest変数に等しい場合のみ赤い〇とする
      fill(255,0,0);
    } else {
      fill(255);
    }
    ellipse(r+d*i,r,d,d);
  }
}

void mousePressed(){
  //マウスがクリックされた位置から一番近い〇を見つける
  //マウスがクリックされた位置と〇の中心座標の最小値を見つける
  float kariKyori = 9999.0; //最小値を見つけるためあり得ない大きさをセット
  int i_min = 0;
  for(int i=0; i<width/d; i++){
    float kyori = dist(r+d*i,r,mouseX, mouseY);
    if(kyori < kariKyori){
      kariKyori = kyori;
      i_min = i;
    }
  }
  nearest = i_min;
}
```

V-2

```
int[] d = {70,40,30,60,50};
int t = 10;
int x = 0;
int y = 200;

void setup(){
  size(600,400);
}

void draw(){
  x = 0;
  background(0);
  for(int i=0; i<d.length; i++){
    x = x + d[i] + t;
    ellipse(x,y,d[i] * 2,d[i] * 2);
    x = x + d[i];
  }
}

void keyPressed(){
  if(key == 'a'){
    //昇順の処理を作成
    for(int i=0;i<d.length-1;i++){
      for(int j=i+1;j<d.length;j++){
        if(d[i] > d[j]){ //昇順
          int tmp = d[i]; //x[i]とx[j]を入れ替えるため一旦退避
          d[i] = d[j];
          d[j] = tmp;
        }
      }
    }
  }
  if(key == 'd'){
    //降順の処理を作成
    for(int i=0;i<d.length-1;i++){
      for(int j=i+1;j<d.length;j++){
        if(d[i] < d[j]){ //降順
          int tmp = d[i]; //x[i]とx[j]を入れ替えるため一旦退避
          d[i] = d[j];
          d[j] = tmp;
        }
      }
    }
  }
}
```