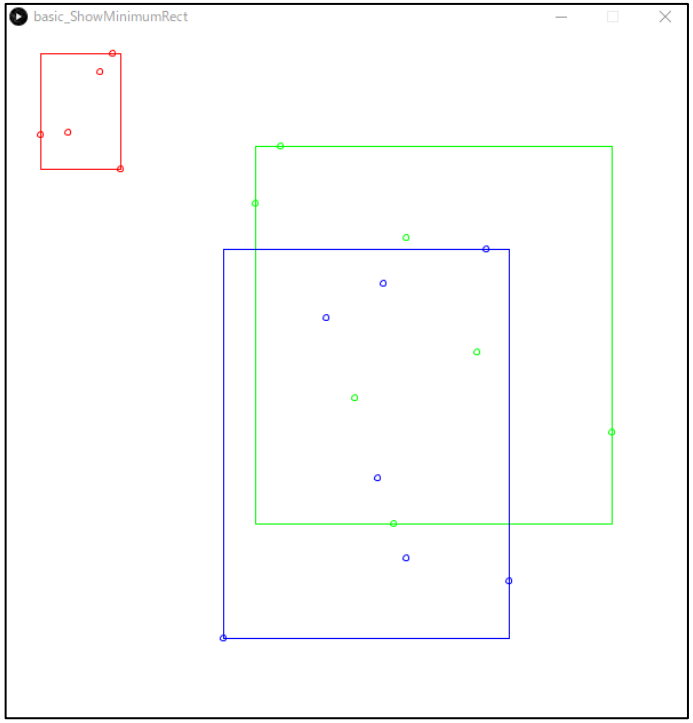
プログラミング演習I （第11回） 課題

- **基本①スケッチ名：basic_ShowMinimumRect**
  - ある点の座標を(x[i], y[i])と表現する点集合を考える
  - この点集合を引数として与えたときに，その点群を囲う最小の四角形（中を塗りつぶさないもの）を描画する関数drawMinimumRectを作成したい。配布したプログラムの関数を完成させなさい。
  - 下図が出力例となるので確認せよ



プログラミング演習I（第11回） 課題

- **基本②：basic_TwinPrime**
  - 双子素数とは、2つの素数の差が2のものをい指す。この双子素数を求めるプログラムを作成したい。
  - まず引数として入力された数値が素数かどうかを判定する関数 isPrime を作成せよ
    - ただし，素数の場合はtrue，素数でない場合はfalseを返り値として返すようにせよ
  - 次に，素数を判定する関数 isPrime を利用して，1000までの双子素数を下記のようにすべて標準出力するとともに，その総数を出力するプログラムを作成せよ．
  - 出力においては下記のルールを守るようにしてください．
    - 下記の数値を決め打ちで出力は不可
    - ペアとなる素数は丸かっこ内にカンマ区切りで書く
    - 余分にスペースが入るのはOKとします

```
(3, 5)
(5, 7)
(11, 13)
(17, 19)
(29, 31)
(41, 43)
(59, 61)
(71, 73)
(101, 103)
(107, 109)
(137, 139)
(149, 151)
(179, 181)
(191, 193)
(197, 199)
(227, 229)

   :
   :

1000までの双子素数の数は**個です
```

プログラミング演習I（第11回） 課題

- **基本③：basic_CosSimilarity**

  ベクトルの類似を計算する方法として、2つのベクトルがなす角度を利用するコサイン類似度というものがある。

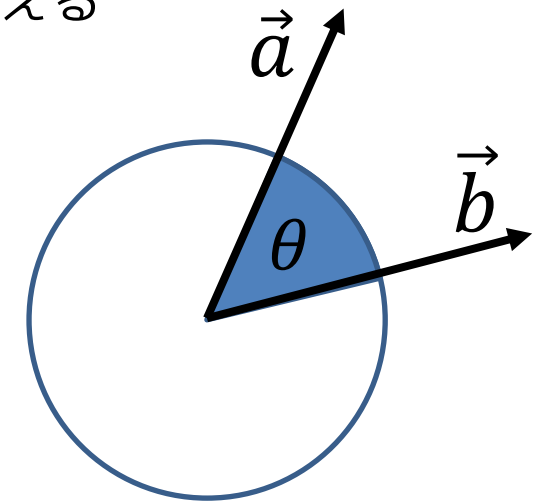
  これは、あるベクトル $\vec{a}$ と $\vec{b}$ の内積が

  $$\vec{a} \cdot \vec{b} = |\vec{a}| \left|\vec{b}\right| cos\theta$$

  で求められることを利用したものであり、

  $$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \left|\vec{b}\right|}$$

  の値が、コサイン類似度と呼ばれるものとなる。つまり、1に近いほど類似しており、-1に近いほど類似していないと言える



プログラミング演習I （第11回） 課題

2つのベクトルが下記で定義されている時

$$\vec{a} = (a_1, a_2, \ldots a_n) \text{ と } \vec{b} = (b_1, b_2, \ldots b_n)$$

そのコサイン類似度は下記の式で求めることができる

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \left|\vec{b}\right|} = \frac{a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} \cdot \sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}}$$

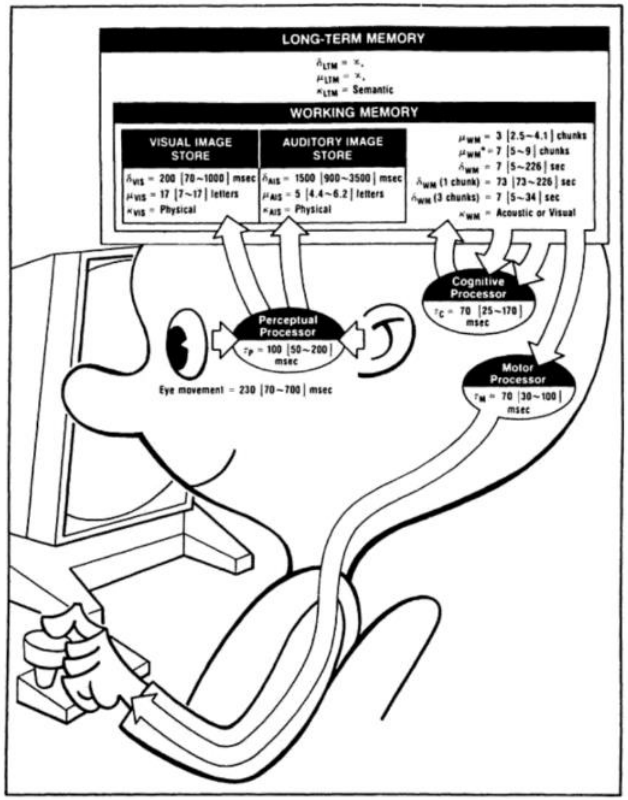
つまり、2つのベクトルの対応する要素の積の和を、各ベクトルの長さ（ノルム）の積で割ったものが、コサイン類似度となる

そこで、ある配列をベクトルとして扱う時、そのベクトルの長さを求める `calcNorm` と、内積を求める `calcInnerProduct` を完成させ、コサイン類似度を求めるプログラムを完成させよ

すべて出力が `true` となったら成功

プログラミング演習I（第11回） 課題

- **発展①スケッチ名：advanced_HCI**
  - ランプが光ったことに反応してボタンを押すまでの時間は少なくとも240ミリ秒かかることが知られている［Card 1983］
  - そこで、まず画面の背景を黒色にし、そこからランダムな待ち時間ののち、画面を白色にし、その白色にしてからユーザがクリックするとまた画面の背景を黒色にせよ。
  - また、その画面を白色にしてから、クリックするまでの時間をミリ秒単位で計測せよ
  - これを10回実施し、最短時間、最長時間、平均時間、分散を計算してコンソールに標準出力するプログラムを作成せよ。



時間を計測するテクニック再

millis()でミリ秒単位の経過時間を取得する

- アプリケーションが起動されてからの時間は millis() で取得することが可能

```
int iStartMillis;
boolean bFlagStart = false; // スタートしたかどうかのフラグ
void setup() {
  size(300, 150);
  fill( 0 ); // 文字色を黒色に設定
}
void draw() {
  background( 255 );
  if( bFlagStart ){ // スタートしていたら～
    text( millis()-iStartMillis, 20, 90 ); // 差分で経過時間を表示
  }
}
void mousePressed(){
  bFlagStart = true; // クリックされたらスタートフラグを立てる
  iStartMillis = millis(); // スタートの経過時間をセット
}
```

プログラミング演習I（第11回） 課題

- **発展②スケッチ名：advanced_Collatz**
  - コラッツ予想とは、下記のルールに従うとすべての自然数が最終的に1になるのではという予想である。まだ本予想は証明されていないが、かなり大きな数まで反例がないことが確かめられている。
  - ルールは下記のとおりである
    - ある数が偶数なら2で割る
    - ある数が奇数なら3を掛けて1を足す
    - 計算結果が1になるまで上記を繰り返す
  - ここで、2から100までの数字において、すべての値の変化を下のように標準出力せよ。また、その最後にステップ数も表示せよ。

```
[2]->1 (1 step)
[3]->10->5->16->8->4->2->1 (7 steps)
[4]->2->1 (2 steps)
[5]->16->8->4->2->1 (5 steps)
[6]->3->10->5->16->8->4->2->1 (8 steps)
[7]->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (16 steps)
[8]->4->2->1 (3 steps)
[9]->28->14->7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (19 steps)
[10]->5->16->8->4->2->1 (6 steps)
[11]->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (14 steps)
[12]->6->3->10->5->16->8->4->2->1 (9 steps)
[13]->40->20->10->5->16->8->4->2->1 (9 steps)
[14]->7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (17 steps)
[15]->46->23->70->35->106->53->160->80->40->20->10->5->16->8->4->2->1 (17 steps)
[16]->8->4->2->1 (4 steps)
```