```
//Vector Library
//CSCI 5611 Vector3 Library
// Stephen J. Guy <sjguy@umn.edu>

public class Vec2 {
  public float x, y;

  public Vec2(float x, float y) {
    this.x = x;
    this.y = y;
  }

  public String toString() {
    return "(" + x + "," + y + ")";
  }

  public float length() {
    return sqrt(x * x + y * y);
  }

  public float lengthSqr() {
    return x * x + y * y;
  }

  public Vec2 plus(Vec2 rhs) {
    return new Vec2(x + rhs.x, y + rhs.y);
  }

  public void add(Vec2 rhs) {
    x += rhs.x;
    y += rhs.y;
  }

  public Vec2 minus(Vec2 rhs) {
    return new Vec2(x - rhs.x, y - rhs.y);
  }

  public void subtract(Vec2 rhs) {
    x -= rhs.x;
    y -= rhs.y;
  }

  public Vec2 times(float rhs) {
    return new Vec2(x * rhs, y * rhs);
  }
  public void mul(float rhs) {
    x *= rhs;
    y *= rhs;
  }

  public void clampToLength(float maxL) {
    float magnitude = sqrt(x * x + y * y);
    if (magnitude > maxL) {
      x *= maxL / magnitude;
      y *= maxL / magnitude;
    }
  }

  public void setToLength(float newL) {
    float magnitude = sqrt(x * x + y * y);
    x *= newL / magnitude;
    y *= newL / magnitude;
  }

  public void normalize() {
    float magnitude = sqrt(x * x + y * y);
    x /= magnitude;
    y /= magnitude;
  }

  public Vec2 normalized() {
    float magnitude = sqrt(x * x + y * y);
    return new Vec2(x / magnitude, y / magnitude);
  }

  public float distanceTo(Vec2 rhs) {
    float dx = rhs.x - x;
    float dy = rhs.y - y;
    return sqrt(dx * dx + dy * dy);
  }
}

Vec2 interpolate(Vec2 a, Vec2 b, float t) {
  return a.plus((b.minus(a)).times(t));
}

float interpolate(float a, float b, float t) {
  return a + ((b - a) * t);
}
```

```
float dot(Vec2 a, Vec2 b) {
  return a.x * b.x + a.y * b.y;
}

// 2D cross product is a funny concept
// ...its the 3D cross product but with z = 0
// ... (only the resulting z component is not zero so we just store it
as a scalar)
float cross(Vec2 a, Vec2 b) {
  return a.x * b.y - a.y * b.x;
}

Vec2 projAB(Vec2 a, Vec2 b) {
  return b.times(a.x * b.x + a.y * b.y);
}

Vec2 perpendicular(Vec2 a) {
  return new Vec2(-a.y, a.x);
}



public class Vec3 {
  public float x, y, z;

  public Vec3(float x, float y, float z) {
    this.x = x;
    this.y = y;
    this.z = z;
  }

  public String toString() {
    return "(" + x + "," + y + "," + z + ")";
  }

  public float length() {
    return sqrt(x * x + y * y + z * z);
  }

  public float lengthSqr() {
    return x * x + y * y + z * z;
  }

  public Vec3 plus(Vec3 rhs) {
    return new Vec3(x + rhs.x, y + rhs.y, z + rhs.z);
  }

  public void add(Vec3 rhs) {
    x += rhs.x;
    y += rhs.y;
    z += rhs.z;
  }

  public Vec3 minus(Vec3 rhs) {
    return new Vec3(x - rhs.x, y - rhs.y, z - rhs.z);
  }

  public void subtract(Vec3 rhs) {
    x -= rhs.x;
    y -= rhs.y;
    z -= rhs.z;
  }

  public Vec3 times(float rhs) {
    return new Vec3(x * rhs, y * rhs, z * rhs);
  }

  public void mul(float rhs) {
    x *= rhs;
    y *= rhs;
    z *= rhs;
  }

  public void clampToLength(float maxL) {
    float magnitude = sqrt(x * x + y * y + z * z);
    if (magnitude > maxL) {
      x *= maxL / magnitude;
      y *= maxL / magnitude;
      z *= maxL / magnitude;
    }
  }

  public void setToLength(float newL) {
    float magnitude = sqrt(x * x + y * y + z * z);
    x *= newL / magnitude;
    y *= newL / magnitude;
    z *= newL / magnitude;
  }
  public void normalize() {
    float magnitude = sqrt(x * x + y * y + z * z);
    x /= magnitude;
    y /= magnitude;
    z /= magnitude;
  }

  public Vec3 normalized() {
    float magnitude = sqrt(x * x + y * y + z * z);
    return new Vec3(x / magnitude, y / magnitude, z / magnitude);
  }

  public float distanceTo(Vec3 rhs) {
    float dx = rhs.x - x;
    float dy = rhs.y - y;
    float dz = rhs.z - z;
    return sqrt(dx * dx + dy * dy + dz * dz);
  }

  public float dot(Vec3 a, Vec3 b) {
    return a.x * b.x + a.y * b.y + a.z * b.z;
  }

}

public float dot(Vec3 a, Vec3 b) {
    return a.x * b.x + a.y * b.y + a.z * b.z;
}
```