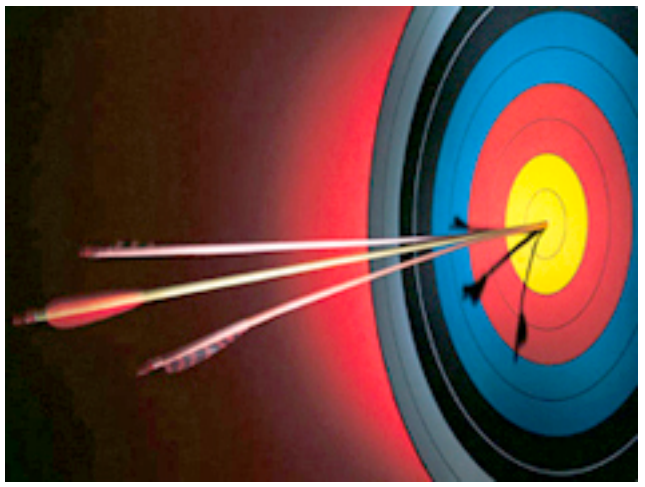
Robin Hood

Intro

We are in a competition to win the archery contest in Sherwood. With our bow and arrows we shoot on a target and try to hit as close as possible to the center.

The center of the target is represented by the values (0, 0) on the coordinate axes.



Goals:

- data structures: lists, sets, tuples
- logical operators: if-elif-else
- loop: while/for
- minimum (optional sorting)

Description:

In the 2-dimensional space, a point can be defined by a pair of values that correspond to the horizontal coordinate (x) and the vertical coordinate (y). The space can be divided into 4 zones (quadrants): Q1, Q2, Q3, Q4. Whose single point of union is the point (0, 0).

If a point is in Q1 both its x coordinate and the y are positive. I leave a link to wikipedia to familiarize yourself with these quadrants.

https://en.wikipedia.org/wiki/Cartesian_coordinate_system (https://en.wikipedia.org/wiki/Cartesian_coordinate_system)

https://en.wikipedia.org/wiki/Euclidean_distance (https://en.wikipedia.org/wiki/Euclidean_distance)

Shots

```
points = [(4, 5), (-0, 2), (4, 7), (1, -3), (3, -2), (4, 5),
          (3, 2), (5, 7), (-5, 7), (2, 2), (-4, 5), (0, -2),
          (-4, 7), (-1, 3), (-3, 2), (-4, -5), (-3, 2),
          (5, 7), (5, 7), (2, 2), (9, 9), (-8, -9)]
```

Tasks

1. Robin Hood is famous for hitting an arrow with another arrow. Did you get it?
2. Calculate how many arrows have fallen in each quadrant.
3. Find the point closest to the center. Calculate its distance to the center.
4. If the target has a radius of 9, calculate the number of arrows that must be picked up in the forest.

In [36]:

```
# Variables

points = [(4, 5), (-0, 2), (4, 7), (1, -3), (3, -2), (4, 5),
          (3, 2), (5, 7), (-5, 7), (2, 2), (-4, 5), (0, -2),
          (-4, 7), (-1, 3), (-3, 2), (-4, -5), (-3, 2),
          (5, 7), (5, 7), (2, 2), (9, 9), (-8, -9)]
```

In [37]:

```
# 1. Robin Hood is famous for hitting an arrow with another arrow. Did you get
it?

for x in points:
    for y in points:
        if x == y:
            is_it_true = True

print(is_it_true)
```

```
True
```

Expected output:

```
True
```

In [38]:

```
# 2. Calculate how many arrows have fallen in each quadrant.

q1 = 0
q2 = 0
q3 = 0
q4 = 0
for x in points:
    if x[0] > 0 and x[1] > 0:
        q1 += 1
    elif x[0] < 0 and x[1] > 0:
        q2 += 1
    elif x[0] < 0 and x[1] < 0:
        q3 += 1
    elif x[0] > 0 and x[1] < 0:
        q4 += 1
print(q1,q2,q3,q4)
```

```
10 6 2 2
```

Expected output:

```
(10, 6, 2, 2)
```

In [43]:

```
# 3. Find the point closest to the center. Calculate its distance to the cente
r
# Defining a function that calculates the distance to the center can help.

import math

def distance_to_center(x,y):
    distance_to_center = math.sqrt((x**2)+(y**2))
    return distance_to_center

closest_point = (float('inf'),float('inf'))
closest_point_distance = 1000


for x in points:
    if distance_to_center(x[0],x[1]) < closest_point_distance:
        closest_point_distance = distance_to_center(x[0],x[1])
        closest_point = x

print(closest_point, closest_point_distance)
```

```
(0, 2) 2.0
```

Expected output:

```
(0, 2)
2.0
```

In [47]:

```
# 4. If the target has a radius of 9, calculate the number of arrows that
# must be picked up in the forest.

picked_up_arrows = []
for x in points:
    if distance_to_center(x[0],x[1]) >= 9:
        picked_up_arrows.append(x)

number_of_picked_up_arrows = int(len(picked_up_arrows))

print (picked_up_arrows, number_of_picked_up_arrows)
```

```
[(9, 9), (-8, -9)] 2
```

Expected output:

```
[(9, 9), (-8, -9)]
2
```