PROJECT DESCRIPTION

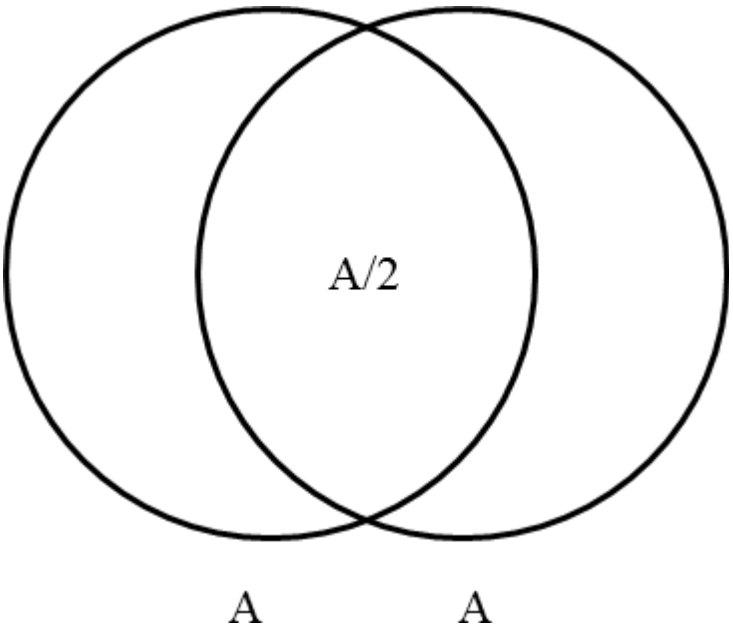
1. INTRODUCTION

For many, a cold beverage in a glass is among the means for survival during summer. The main purpose of a coaster is to absorb condensation dripping along the glass, and thereby protect the surface of a table or any other surface used for placing the glass. The coasters are also of interest to **tegestologists**.

For the sake of reference, the project as well as its product is called **CHEERS**.

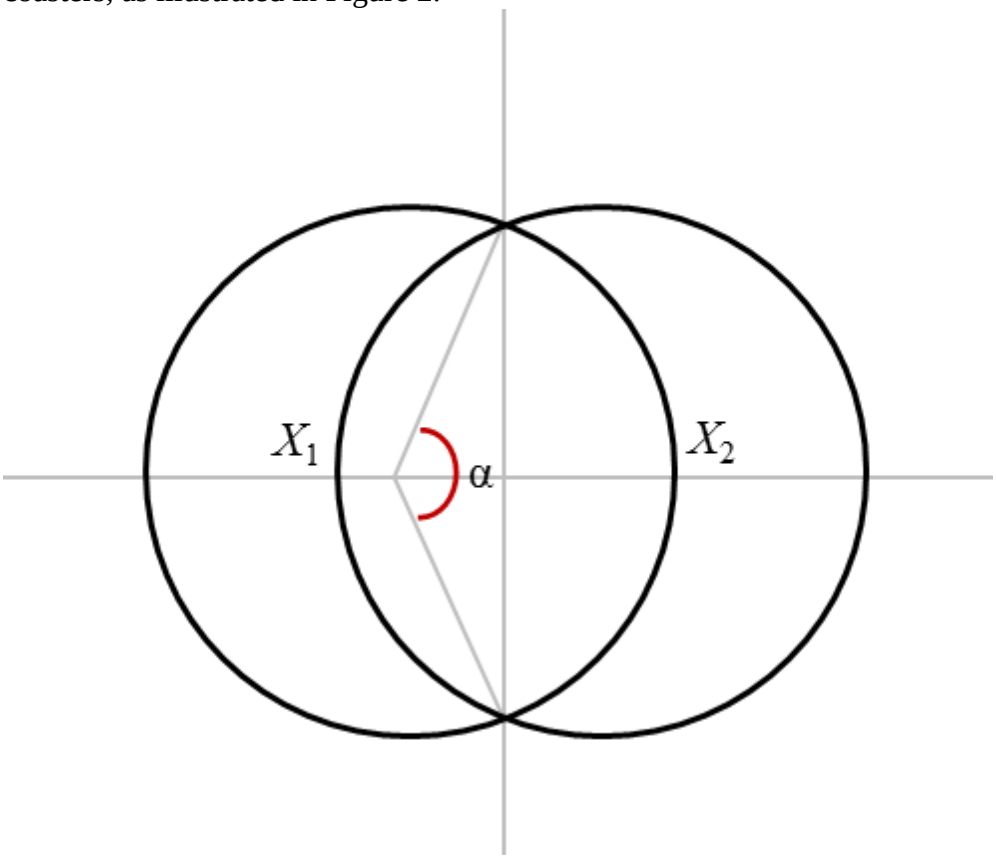
2. PROBLEM

Let there be two circular coasters of equal area (and negligible height). The purpose of **CHEERS** is to find how far the two coasters need to be moved on top of each other such that the area of the overlapping region is half the area of any one of the coasters, as illustrated in Figure 1.



3. SOLUTION

The segment $X1X2$ on the x-axis is an indicator of the degree of overlap between the two coasters, as illustrated in Figure 2.



Let R be the radius of any one of the coasters, and let α be the angle with vertex at the center of the left circle, as shown in Figure 2.
Then the length l of the segment $X1X2$ is given by the equation
$l = 2R(1 - \cos(\alpha/2))$,
and α is given by the equation
$\alpha - \sin(\alpha) = \pi/2$.

4. CHEERS

The purpose is to compute l. (This, in turn, requires computing α.)
The work on **CHEERS** has been divided into an **interrelated collection of problems** to be solved. Each problem has a note associated with it, meant to serve as a guide for scoping, understanding, and/or solving the problem.

PROBLEM 1.

Provide an outline of the solution, including the object-oriented design corresponding to **program**. **You must** construct a **CRC card model**.

NOTE

For each CRC card, the following must be described: (a) role, (b) responsibilities and the rationale for the responsibilities, and (c) collaborations and reasons for collaborating.

PROBLEM 2.

Rationalize the selection of each algorithm deployed in **source code**, and document the algorithms using **pseudocode** (as well as natural language, if necessary).

NOTE
The pseudocode should not be too close to the implementation.

PROBLEM 3.

S must be written in a manner that seeks as many **opportunities for reuse** as possible. **S** could, for example, make use of application programming interfaces or library functions, available **natively or otherwise**.

NOTE

you must provide the steps taken towards making **source code** to be **readable, modifiable, testable, and understandable**. **You must** provide the steps taken towards making the corresponding **program** to be **general, robust, and usable**.

You must provide a sample output, say, for different values of *R*. The **output** of **program** must be expressed in XML. This output must be valid with respect to some XML DTD.

Source code must be written in **Python**.

You must use a debugger.

Source code must not depend on any particular development environment (such as a specific operating system or an IDE). (**you must** prove that **Source code** is independent of any IDE.)
Source code must be **documented appropriately** using a **'standard' documentation system**. (For Python, such a documentation system is Pydoc.) It is recommended that the internal documentation and the source code **evolve synchronously**.

Source code must not violate established **principles of programming**. (For example, such

principles for **object-oriented programming** are those related to abstraction, encapsulation, inheritance, and/or polymorphism.)

Source code must not declare π as a named constant. (It must compute the value of π.)

Source code must have proper support for **handling exceptions**.

Source code must have proper **error messages**.

You must provide a listing of **Source code**.

You must provide a description of instructions for processing **Source code** (by compiling or interpreting it, as the case may be).