Name:

YOU MAY READ THIS PAGE BEFORE THE EXAM BEGINS

1. You have 75 minutes to create and submit the 2 programs in this exam.
2. When instructed to start, download and extract the following PyCharm Project under the `~/workspace` folder if you do not have it there already.

   `https://www.cs.umb.edu/~siyer/teaching/ipp.zip`

3. Open the Project in PyCharm. To create a program, right-click on `ipp` in the top-left window and then select the *New → Python File* menu. Enter the name of the program in the pop-up window. Note that the name is case-sensitive and must match the suggested name exactly.
4. You may use the text, your notes, your code from the assignments, and the code on the CS110 course website. No form of communication is permitted (eg, talking, texting, etc.) during the exam, except with the course staff.
5. Submit your programs (`.py` files) on Gradescope under the assignment named `Sample Programming Exam 2`.
6. Return this exam sheet to the course staff with your name written at the top. Failing to do so will void your exam submission on Gradescope.
7. You are *not* allowed to leave the exam hall before the official end time even if you are done early.
8. Your programs will be graded based on correctness, clarity, and efficiency.
9. Discussing the exam contents with anyone who has not taken the exam is a violation of the academic honesty code.

DO NOT READ FURTHER UNTIL SO INSTRUCTED

Problem 1. (18 points) Implement a data type called `Circle` that represents a circle of radius r centered at (h, k) and supports the following API:

circle.Circle

`Circle(h, k, r)`	constructs a circle c of radius r centered at (h, k); when no arguments are given, c is a unit circle centered at the origin
`c.area()`	returns the area of c, calculated as πr^2
`c.circumference()`	returns the circumference of c, calculated as $2\pi r$
`c.contains(x, y)`	returns `True` if c contains† (x, y) and `False` otherwise
`c.scale(r)`	returns a new circle whose center is the same as c, but with radius r
`c.distanceTo(d)`	returns the Euclidean distance‡ between the centers of circles c and d
`c == d`	returns `True` if c and d denote the same circle (ie, have the same centers and radii), and `False` otherwise
`c < d`	returns `True` if the area of `c` is smaller than the area of `d` and `False` otherwise
`str(c)`	returns a string representation of c, as `(h, k, r)`

† A point (x, y) is contained in a circle of radius r centered at (h, k) if $(h - x)^2 + (k - y)^2 \leq r^2$.

‡ The Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Enter the following starter code in the program and replace the ellipsis (...) with your code.

circle.py

```
import math

class Circle:
    def __init__(self, h=0, k=0, r=1):
        ...

    def area(self):
        ...

    def circumference(self):
        ...

    def contains(self, x, y):
        ...

    def scale(self, r):
        ...

    def distanceTo(self, other):
        ...

    def __eq__(self, other):
        ...

    def __lt__(self, other):
        ...

    def __str__(self):
        ...

# Unit tests the data type.
def _main():
    import stdio

    c = Circle()
    d = Circle(1, 1, 2)
    stdio.writeln(c)
    stdio.writeln(d)
    stdio.writeln(c.area())
    stdio.writeln(c.circumference())
    stdio.writeln(c.contains(1, 1))
    stdio.writeln(c.scale(2))
    stdio.writeln(c.distanceTo(d))
    stdio.writeln(c == d or c == c)
    stdio.writeln(d < c)

if __name__ == "__main__":
    _main()
```

```
>_ ~/workspace/dummy_project
$ python3 circle.py
(0, 0, 1)
(1, 1, 2)
3.141592653589793
6.283185307179586
False
(0, 0, 2)
1.4142135623730951
True
False
```

Problem 2. (7 points) Implement a program called `filter.py` that does the following:

i. reads floats h, k, and r as command-line arguments;

ii. creates a `Circle` object c of radius r centered at (h, k);

iii. reads (x, y) points (both floats) from standard input until EOF (use the functions `stdio.isEmpty()` and `stdio.readFloat()`); and

iv. writes the fraction of points that are *not* contained in c.

Enter the following starter code in the program and replace the ellipsis (...) with your code.

filter.py

```
from circle import Circle
import stdio
import sys

# Entry point.
def main():
    ...

if __name__ == "__main__":
    main()
```

```
>_ ~/workspace/dummy_project
$ python3 filter.py 0 0 1
0 0 2 2 0.5 0.5 <ctrl-d>
0.6666666666666666
```

Files to Submit

1. `circle.py`
2. `filter.py`

Answers

Problem 1.

circle.py

```python
import math


class Circle:
    # Constructs a circle of radius r centered at (h, k); when no arguments are given, the circle
    # is a unit circle centered at the origin.
    def __init__(self, h=0, k=0, r=1):
        self._h = h
        self._k = k
        self._r = r

    # Returns the area of this circle.
    def area(self):
        return math.pi * self._r ** 2

    # Returns the circumference of this circle.
    def circumference(self):
        return 2 * math.pi * self._r

    # Returns True if this circle contains the point (x, y) and False otherwise.
    def contains(self, x, y):
        return (self._h - x) ** 2 + (self._k - y) ** 2 <= self._r ** 2

    # Returns a new circle whose center is the same as this circle, but with radius r.
    def scale(self, r):
        return Circle(self._h, self._k, r)

    # Returns the Euclidean distance between the centers of this circle and other.
    def distanceTo(self, other):
        return ((self._h - other._h) ** 2 + (self._k - other._k) ** 2) ** 0.5

    # Returns True if this circle is the same as other (ie, they have the same centers and radii)
    # and False otherwise.
    def __eq__(self, other):
        return self._h == other._h and self._k == other._k and self._r == other._r

    # Returns True if this circle's area is smaller than other's area and False otherwise.
    def __lt__(self, other):
        return self.area() < other.area()

    # Returns a string representation of this circle.
    def __str__(self):
        return "(" + str(self._h) + ", " + str(self._k) + ", " + str(self._r) + ")"


# Unit tests the data type.
def _main():
    import stdio

    c = Circle()
    d = Circle(1, 1, 2)
    stdio.writeln(c)
    stdio.writeln(d)
    stdio.writeln(c.area())
    stdio.writeln(c.circumference())
    stdio.writeln(c.contains(1, 1))
    stdio.writeln(c.scale(2))
    stdio.writeln(c.distanceTo(d))
    stdio.writeln(c == d or c == c)
    stdio.writeln(d < c)


if __name__ == "__main__":
    _main()
```

Problem 2.

filter.py

```python
from circle import Circle
import stdio
import sys
```

```
4
5
6   # Entry point.
7   def main():
8       # Accept h (float), k (float), and r (float) as command-line arguments.
9       h = float(sys.argv[1])
10      k = float(sys.argv[2])
11      r = float(sys.argv[3])
12
13      # Create a Circle object c with parameters (h, k, r).
14      c = Circle(h, k, r)
15
16      inside = 0  # number of points inside the circle.
17      total = 0  # total number of points.
18
19      # Until standard input is not empty...
20      while not stdio.isEmpty():
21          # Read floats x and y.
22          x = stdio.readFloat()
23          y = stdio.readFloat()
24
25          # Increment total by 1.
26          total += 1
27
28          # If the circle c contains (x, y), increment inside by 1.
29          if c.contains(x, y):
30              inside += 1
31
32      # Write the fraction of points contained in the circle c.
33      stdio.writeln(inside / total)
34
35
36  if __name__ == "__main__":
37      main()
```