Name:

YOU MAY READ THIS PAGE BEFORE THE EXAM BEGINS

1. You have 75 minutes to create and submit the 3 programs in this exam.
2. When instructed to start, download and extract the Dummy PyCharm Project under the `~/workspace` folder if you do not have it there already.

   `https://www.cs.umb.edu/~siyer/teaching/cs110/dummy_project.zip`

3. Open the Dummy Project in PyCharm. To create a program, right-click on `dummy_project` 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 projects, 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 1"`.
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 (including comments), design, 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 OR DOWNLOAD THE DUMMY PROJECT UNTIL SO INSTRUCTED

Problem 1. (5 Points) Create a program called `distance.py` that takes four floats x_1, y_1, x_2, and y_2 as command-line arguments representing the Cartesian coordinates of two points (x_1, y_1) and (x_2, y_2) on a plane, and writes the Euclidean distance between the points, calculated as $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Enter the following starter code in the program and replace the ellipsis (...) with your code.

distance.py

```
import math
import stdio
import sys

...
```

~/workspace/dummy_project

```
$ python3 distance.py 1.0 2.0 4.0 6.0
5.0
```

Problem 2. (10 Points) Create a program called `minmax.py` that accepts integers from standard input and writes their minimum and maximum values to standard output. Enter the following starter code in the program and replace the ellipsis (...) with your code.

minmax.py

```
import stdio

...
```

~/workspace/dummy_project

```
$ python3 minmax.py
22 21 -6 -95 -70 -48 -19 -76 73 -31
<ctrl-d>
Minimum = -95
Maximum = 73
```

Relevant standard input functions: `stdio.isEmpty()`, `stdio.readInt()`.

Problem 3. (10 Points) Create a library called `listops.py` that supports the following API:

listops

`find(a, k)`	returns the index of the *first* occurrence of the element k within the list a, and -1 if the element does not exist in the list
`rfind(a, k)`	returns the index of the *last* occurrence of the element k within the list a, and -1 if the element does not exist in the list
`count(a, k)`	returns the number of occurrences of the element k within the list a
`contains(a, k)`	returns `True` if the list a contains the element k, and `False` otherwise

Enter the following starter code in the library and replace the ellipses (...) with your code.

listops.py

```
def find(a, k):
    ...

def rfind(a, k):
    ...

def count(a, k):
    ...

def contains(a, k):
    ...

# Unit tests the library.
def _main():
    import stdio
```

```
    a = [5, -1, 3, 4, 3, -2, 3, 5]
    stdio.writeln(find(a, 3))
    stdio.writeln(rfind(a, 3))
    stdio.writeln(count(a, 3))
    stdio.writeln(contains(a, 6))

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

```
>_ ~/workspace/dummy_project
$ python3 listops.py
2
6
3
False
```

1. `distance.py`
2. `minmax.py`
3. `listops.py`

Answers

Problem 1.

distance.py

```python
import math
import stdio
import sys

# Accept x1 (float), y1 (float), x2 (float), and y2 (float) as command-line arguments.
x1 = float(sys.argv[1])
y1 = float(sys.argv[2])
x2 = float(sys.argv[3])
y2 = float(sys.argv[4])

# Set distance to the Euclidean distance between points (x1, y1) and (x2, y2).
distance = math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)

# Write distance to standard output.
stdio.writeln(distance)
```

Problem 2.

minmax.py

```python
import stdio

# Set minValue and maxValue to None.
minValue = None
maxValue = None

# Until standard input is empty...
while not stdio.isEmpty():
    # Read an integer x.
    x = stdio.readInt()

    # Update minValue to x if x is smaller and leave it unchanged otherwise.
    minValue = x if minValue == None or x < minValue else minValue

    # Update maxValue to x if x is bigger and leave it unchanged otherwise.
    maxValue = x if maxValue == None or x > maxValue else maxValue

# Write minValue and maxValue to standard output.
stdio.writeln("Minimum = " + str(minValue))
stdio.writeln("Maximum = " + str(maxValue))
```

Problem 3.

listops.py

```python
# Returns the index of the first occurrence of the element k within the list a, and -1 if the
# element does not exist in the list.
def find(a, k):
    # For each index i of a...
    for i in range(len(a)):
        # If the ith value in a equals k, return its index i.
        if a[i] == k:
            return i

    # k does not exist in a, so return -1.
    return -1

# Returns the index of the last occurrence of the element k within the list a, and -1 if the
# element does not exist in the list.
def rfind(a, k):
    # For each index i of a in reverse...
    for i in reversed(range(len(a))):
        # If the ith value in a equals k, return its index i.
        if a[i] == k:
            return i

    # k does not exist in a, so return -1.
    return -1

# Returns the number of occurrences of the element k within the list a.
```

```python
def count(a, k):
    # Set count (number of occurrences of k in a) to 0.
    count = 0

    # For each value v in a...
    for v in a:
        # If v equals k, increment count by 1.
        if v == k:
            count += 1

    # Return count.
    return count

# Returns True if the list a contains the element k, and False otherwise.
def contains(a, k):
    # For each value v in a...
    for v in a:
        # If v equals k, return true.
        if v == k:
            return True

    # k does not exist in a, so return False.
    return False

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

    a = [5, -1, 3, 4, 3, -2, 3, 5]
    stdio.writeln(find(a, 3))
    stdio.writeln(rfind(a, 3))
    stdio.writeln(count(a, 3))
    stdio.writeln(contains(a, 6))

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