CSC263 – Problem Set 1

Remember to write your **full name** and **student number** prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, **clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted**.

Remember that you are required to submit your problem sets as both LaTeX `.tex` source files and `.pdf` files. There is a 10% penalty on the assignment for failing to submit both the `.tex` and `.pdf`.

Due Jan 23, 2025, 22:00; required files: ps1.pdf, ps1.tex, ps1.py

Answer each question completely, always justifying your claims and reasoning. Your solution will be graded not only on correctness, but also on clarity. Answers that are technically correct but are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

You may work in groups of up to TWO to complete these questions.

1. **[12 points]** Consider the following algorithm, where `locations` is a list of geographic coordinates, and `radius` specifies the range of interest in kilometers.

```
1 ProximityCheck(locations, x_q, y_q, radius = 10):
2   for (x, y) in locations:
3     d = \sqrt{(x - x_q)^2 + (y - y_q)^2}
4     if d < radius:
5       print("Within range!")
6     else:
7       print ("Out of range!")
```

Suppose the input list `locations` contains n points distributed uniformly and independently over a square region of side length L. The query point (x_q, y_q) is fixed at the center of the square. Each location is equally likely to fall anywhere within the square.

(a) (1 point) What is the probability that `ProximityCheck` prints "Out of range!" for every point in `locations`? Assume the default value `radius = 10`. Justify your answer carefully: show your work and explain your calculation.

(b) (1 point) What is the probability that `ProximityCheck` prints "Within range!" for all n points? Assume the default value `radius = 10`. Justify your answer carefully: show your work and explain your calculation.

(c) (2 points) What is the expected number of times "Within range!" is printed for $n = 5$ locations? Assume the default value `radius = 10`. Justify your answer carefully: show your work and explain your calculation.

(d) (4 points) Consider the following modified algorithm. Please calculate the worst-case and best-case runtime, as well as their probability.

```
1 ProximityCheck(locations, x_q, y_q, radius):
2   total_distance = 0
3   for (x, y) in locations:
4     d = \sqrt{(x - x_q)^2 + (y - y_q)^2}
5     if d < radius:
6       total_distance += d
7     elif d >= radius:
8       print("Terminating...")
9       return total_distance
10  return total_distance
```

(e) (4 points) What is the expected number of times the for loop runs in the modified algorithm? Show your work and explain your calculation. You don't have to simplify your final answer.

Hint: The probability that a single point is within the circular region of radius r centered at (x_q, y_q) is proportional to the ratio of the circle's area to the square's area:

$$p = \frac{\pi r^2}{L^2}.$$

For multiple independent points, probabilities combine according to the binomial distribution.

Programming Question

The best way to learn a data structure or an algorithm is to code it up. In each problem set, we will have a programming exercise for which you will be asked to write some code and submit it. You may also be asked to include a write-up about your code in the PDF/TEXfile that you submit. Make sure to **maintain your academic integrity** carefully, and protect your own work. The code you submit will be checked for plagiarism. It is much better to take the hit on a lower mark than risking much worse consequences by committing an academic offence.

2. **(12 points)** In this question, we will solve a problem that we call **spy263**. The function `spy263` takes a list of commands that operate on the current collection of data. Your task is to process the commands in order and return the required list of results. There are two kinds of commands: `insert` commands and `find_spy` commands.

   An `insert` command is a string of the form `insert x`, where `x` is an integer. (Note the space between `insert` and `x`.) This command adds `x` to the collection.

   A `find_spy` command is simply the string `find_spy`. It retrieves the $\lceil \phi \times n \rceil$-th smallest element in the collection, where $\phi = 0.263$ is a position where a spy chooses to hide, and n is the current size of the collection.

   Your goal is to implement `insert` in $O(\lg n)$ time worst-case, and `find_spy` in $O(1)$ time worst-case. Your algorithm should also have $O(n)$ space complexity. Here, n is the number of elements currently in the collection. The list returned by `spy263` consists of the results, in order, from each `find_spy` command.

   Let's go through an example. Here is a sample call of `spy_263`:

```
spy263(
  ['insert 15',
   'find_spy',
   'insert 6',
   'insert 2',
   'insert 8',
   'find_spy',
   'insert -5'
   'insert -8'
   'insert 3'
   'insert 20'
   'find_spy',
  ])
```

   These commands corresponds to the following steps:

   - The collection begins empty, with no elements.
   - We insert 15. The collection contains just the integer 15.
   - We then have our first `find_spy` command. The result is the $\lceil \phi \times 1 \rceil = \lceil 0.263 \rceil = 1$st smallest element currently in the collection, which is 15.
   - We insert 6. The collection now contains 15 and 6.
   - We insert 2. The collection now contains 15, 6, and 2.
   - We insert 8. The collection now contains 15, 6, 2, and 8.
   - Now we have our second `find_spy` command. The result is the $\lceil \phi \times 4 \rceil = \lceil 1.052 \rceil = 2$nd smallest element currently in the collection, which is 6.

- We insert -5. The collection now contains 15, 6, 2, 8, and -5.
- We insert -8. The collection now contains 15, 6, 2, 8, -5, and -8.
- We insert 3. The collection now contains 15, 6, 2, 8, -5, -8 and 3.
- We insert 20. The collection now contains 15, 6, 2, 8, -5, -8, 3 and 20.
- Now we have our third and final `find_spy` command. The result is the $\lceil \phi \times 8 \rceil = \lceil 2.104 \rceil = 3$rd smallest element currently in the collection, which is 2.

So, the above call `spy263` returns `[15, 6, 2]`, which are the three values produced by the `find_spy` commands.

Requirements:

- Your code must be written in Python 3, and the filename must be `ps1.py`.
- We will grade only the `spy263` function; please do not change its signature in the starter code. Include as many helper functions as you wish.
- All code that you submit must be your own, including any helper functions.
- Your code must compile and otherwise be testable in order to earn credit for the auto-graded portion of this question.
- Your written explanation and runtime analyses must be reasonable to earn points on the coding components of this question, regardless of your autotest result.
- You can earn up to 12 points if you **only use materials from weeks 1-2, and previous data structures covered in CSC148**
- You can earn up to 6 points if you use other materials.
- For each test-case that your code is tested on, your code must run within 10x the time taken by our solution. Otherwise, your code will be considered to have timed out.

Write-up: in your `ps1.pdf/ps1.tex` files, briefly but clearly describe the main ideas behind your algorithms (3 points). Informally argue why your code is correct, and has the desired runtime (3 points). Your code will be tested via unit tests (6 points)—however, **your written explanation and runtime analyses must be reasonable to earn points on the coding components of this question, regardless of your autotest result.**

Hint: It is up to you to decide what data structure or *combination of data structures* you wish to use to store the collection.