Plagiarism

Below are rules of thumb that the course considers reasonable and not reasonable.

Reasonable

- Communicating with classmates about assessments and properly citing those discussions.
- Discussing the course's material with others to understand it better.
- Helping a classmate identify a bug in their code, as by viewing, compiling, or running their code after you have submitted that portion of the assignment yourself.
- Incorporating a few lines of code that you find online or elsewhere into your own code, provided that those lines are not themselves solutions to assigned work and that you cite the lines' origins.
- Using third-party AI-based software (including ChatGPT, GitHub Copilot, the new Bing, et al.) to suggest snippets of code or to fix bugs, provided that you cite the codes' origins.
- Sending or showing code that you've written to someone, possibly a classmate, so that they might help you identify and fix a bug.
- Turning to the web or elsewhere for instruction beyond the course's own, for references, and for solutions to technical difficulties, but not for outright solutions to assigned work.
- Whiteboarding solutions with others using diagrams or pseudocode but not actual code.
- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you.

Not Reasonable

- Using third-party AI-based software (including ChatGPT, GitHub Copilot, the new Bing, et al.) to generate code without citation.
- Using third-party AI-based software to generate code that you later submit but you cannot explain how the code works.
- Accessing a solution to some assessment prior to (re-)submitting your own.
- Asking a classmate to see their solution to some assessment before (re-)submitting your own.
- Failing to cite (as with comments) the origins of code or techniques that you discover outside of the course's own lessons and integrate into your own work, even while respecting this policy's other constraints.
- Giving or showing to a classmate a solution to an assessment when it is they, and not you, who is struggling to solve it.
- Paying or offering to pay an individual for work that you may submit as (part of) your own.
- Providing or making available solutions to assessments to anyone, whether a past, present, or prospective future student.
- Searching for or soliciting outright solutions to assessments online or elsewhere.
- Splitting an assessment's workload with another individual and combining your work.
- Submitting (after possibly modifying) the work of another individual beyond the few lines allowed herein.
- Viewing another's solution to an assessment and basing your own solution on it.

Instructions:
“””
Every submission requires pseudo-code written out at the top of the file.
Write comments only for code that is **not** self-explanatory.
“””

1. Put all your code into **ONE** file called NAME-CLASS-Number-game.py
2. Commit and push to GitHub

Number game

Create a program that guesses a secret number.

The program will work like this: you think of a whole number.
The number is in the range of 0 up to but not including 100.
The program tries to guess the number you are thinking of.
Next, you answer whether or not this guess is too high, too low, or correct.

Your solution must use bisection search!

Here is an example game:

```
Think of a whole number between 0 and 100
Is it 50?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 75?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 87?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. h
Is it 81?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 84?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. h
Is it 82?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 83?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. c
Game over. It was: 83
```

Hint:

Your initial low should be 0 and initial high should be 100.
What is the first guess? How do you calculate the next guess?

Note: use `input()` to read user commands.

If a user does not type one of `h`, `l`, or `c` **you need to print an error message**.
Next, you should **ask the question again**.

Example:

```
Is it 81?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. g
Invalid input.
Is it 81?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct.
```

Testcase 1. Secret number is 42

```
Think of a whole number between 0 and 100
Is it 50?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. h
Is it 25?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 37?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 43?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. h
Is it 40?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 41?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 42?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. c
Game over. It was: 42
```

Testcase 2. Secret number = 91

```
Think of a whole number between 0 and 100
Is it 50?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 75?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 87?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 93?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. h
Is it 90?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. l
Is it 91?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. y
Invalid input.
Is it 91?
Type 'h' if this guess is too high. Type 'l' if this guess is too low. Type
'c' if this guess is correct. c
Game over. It was: 91
```

Note:
Printing on the same line (in case you need too)

Try the following in your console:

```
# Notice how if we have two print statements
print("Hi")
print("there")

# The output will have each string on a separate line:
Hi
there

# Now try adding the following:
print("Hi",end='')
print("there")
print("Hi",end='*')
print("there")

# The output will place the subsequent string on the same line
# and will connect the two prints with the character given by end
Hithere
Hi*there
```