CS106A Summer 2025
Neel Kishnani

Midterm Exam

1. Copycat Karel	18
2. Reverse Khansole	18
3. Bug Busting a. Revisited b. Exponentiation	12 6 6
4. Graduate Courses	15
5. Sum to Value	16
	79

_____
Print name (**legibly**)

_____
Your SUID (e.g. 006892056)

Exam instructions: There are 5 questions. Write all answers directly on the exam. This printed exam is **closed-book and closed-device**; you may refer only to your one letter-sized page of prepared notes and the provided reference sheet.
Python coding guidelines: Unless otherwise restricted in the instructions for a specific problem, you are free to use any of the libraries, functions, and data types we have learned in class. You don't need `import` statements in your solutions, just assume the required library files are available to you. You are free to create helper functions unless the problem states otherwise. Helper functions can be defined before or after they are used. Comments are not required, but when your code is incorrect, comments could clarify your intentions and help earn partial credit.

THE STANFORD UNIVERSITY HONOR CODE

A. The Honor Code is an undertaking of the students, individually and collectively:

(1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

(2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid as far as practicable, academic procedures that create temptations to violate the Honor Code.

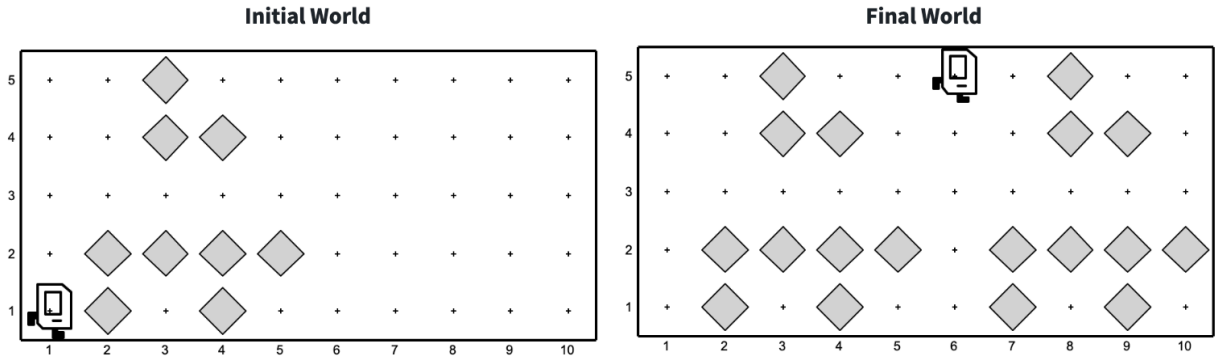
C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work. I acknowledge and accept the Honor Code.

_____(signature)

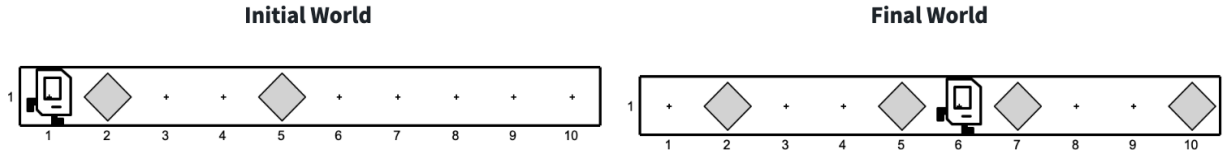
I accept and acknowledge the honor code.

Problem 1: Copycat Karel (18 points)

Write a program that has Karel copy a pattern of beepers. **Karel always starts in position (1, 1) facing East with infinite beepers in its bag.** Avenues (columns) 2 through 5 in the world contain a random pattern of beepers, where each corner on those avenues either has 0 or 1 beeper on it. Karel should copy the pattern of beepers on avenues (columns) 2 through 5 into avenues (columns) 7 through 10. Note that all the corners on avenues 7 thought 10 are all empty to begin with. For example, given the initial world on the left below, Karel's program should result in the final world shown on the right below:



Note that the world can be of arbitrary height (one or more streets/rows) and the pattern of beepers to copy extends to the top of the world, no matter how high the world is. For example, the world may be only one row high, as shown in the example initial/final worlds below:



Your program should work with any world that meets the specifications described in this problem (not just the specific initial worlds pictured above).

Here are a few assumptions you can make in your program:

- The world is always 10 avenues (columns) wide, but may be of arbitrary height (number of rows).
- Other than the beepers that form the pattern in avenues (columns) 2 through 5, there are no other beepers initially placed in the world.
- Other than the border around the world, there are no other walls in the world.
- When the program finishes, there should be no other beepers in the world other than the original pattern in avenues 2 through 5 (which should not be changed) and the copy of the beeper pattern in avenues 7 through 10.
- After Karel's program completes, Karel may be in any location and facing any direction.
- You can only use the functions in the Karel portion of the reference sheet and no additional Python features. Especially note that **you cannot use any variables in your Karel program (other than using a variable `i` as an index variable in `for` loops, if you need to use `for` loops).**

_____ (Your SUID, e.g. 006892456, required on every page)

Please write your solution for Problem 1 here.

```
from karel.stanfordkarel import *









def main():









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

_____ (Your SUID, e.g. 006892456, required on every page)

Use this page as additional space for Problem 1, if needed.

This page intentionally left blank.

Problem 2: Reverse Khansole (18 points)

In Assignment 2 you wrote Khansole Academy, an interactive console program that tests the user's mathematical skills. In this problem, we're flipping the roles: the user will provide math problems for you (the programmer) to solve.

Write a console program that prompts the user for addition and subtraction problems for the program to solve. Parse the input to find the two numbers being added/subtracted and print the correct answer. The program will keep prompting the user until they enter the sentinel value: an empty equation to solve.

You may assume that:

- A non-empty equation will always be of the form **`num1 operation num2`** where **`num1`** and **`num2`** are integers and **`operation`** is a single character. These three portions of the input are guaranteed to be space-separated.
- Your program should handle the operations **`+`** and **`-`**. if the user enters a different operation, inform them that the equation is invalid and re-prompt for a new equation to solve.

Here is a sample run of the program:

```
Enter an equation to solve: 12 + 10
The answer is 22

Enter an equation to solve: 9 - 12
The answer is -3

Enter an equation to solve: 5 * 2
That is an invalid equation

Enter an equation to solve: 30 + 14
The answer is 44

Enter an equation to solve:
```

Here, the user provided 5 equations:

1. A valid addition problem (**`12 + 10`**)
2. A valid subtraction problem (**`9 - 12`**)
3. An invalid problem which doesn't have addition or subtraction
4. A valid addition problem (**`30 + 14`**)
5. The sentinel value: an empty input

Please write your solution for Problem 2 here.

```
def main():
```

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

_____ (Your SUID, e.g. 006892456, required on every page)

Use this page as additional space for Problem 2, if needed.

This page intentionally left blank.

Problem 3: Bug Busting (12 points)

Read the following code snippets and their intended functionality. Both snippets are incorrect as currently written. For each snippet, write what the program currently outputs, and describe the bug(s) in the implementation. Then, write the corrected version of the snippet.

3a. Remove digits, revisited (6 points)

This program attempts to print a new string that contains only the non-numeric characters in **`my_str`**. The expected output is: **`The updated string is CSA`**

```
1           def remove_digits(s):
2               updated = ""
3               for ch in s:
4                   if not ch.isdigit():
5                       updated += ch
6               return updated
7
8           def main():
9               my_str = "CS106A"
10              remove_digits(my_str)
11              print(f"The updated string is {my_str}")
```

What does this program, as currently implemented, print to the console?

Describe the issue(s) with this implementation.

This program can be fixed by updating one line of code. Write the line number and the corrected line of code.

3b: Exponentiation (6 points)

This program attempts to print the result of the number 2 raised to the exponent 3. We expect this to print 8, which is 2 times 2 times 2.

```
1           def perform_exponentiation(exponent, base):
2               print(base ** exponent)
3
4           def main():
5               num1 = 2
6               num2 = 3
7               perform_exponentiation(num1, num2)
```

What does this program, as currently implemented, print to the console?

Describe the issue(s) with this implementation.

This program can be fixed by updating one line of code. Write the line number and the corrected line of code.

Problem 4: Graduate Courses (15 points)

Course names at Stanford are formatted as follows: department abbreviation, followed by the numeric course code, and optionally followed by a letter. Some examples of this format include:

- **`"CS22A"`**: **`"CS"`** is for the Computer Science department, **`"22"`** is the course code, and the optional **`"A"`** is to differentiate the class from CS22B.
- **`"EARTHSYS101"`**: **`"EARTHSYS"`** is for the Earth Systems department, and **`"101"`** is the course code.
- **`"AFRICAAM217"`**: **`"AFRICAAM"`** is for the African and African American Studies department, and **`"217"`** is the course code.

The university defines a graduate-level course as one in which the **course code meets both of these requirements**:

1. At least 3 digits
2. Starts with a number greater than or equal to 2

So, for our examples above:

- **`"CS22A"`** would not be a graduate course since its course code is less than 3 digits.
- **`"EARTHSYS101"`** would not be a graduate course since its course code starts with the number 1.
- **`"AFRICAAM217"`**would be a graduate course since its course code is 3 digits and starts with the number 2.

Here is that same logic, described in Python doctests:

```
>>> is_graduate_course("CS22A")
False

>>> is_graduate_course("EARTHSYS101")
False

>>> is_graduate_course("AFRICAAM217")
True
```

Write a function **`is_graduate_course`** which takes in a string **`course`** representing a course name as a parameter and returns a boolean value representing if the course is graduate-level or not.

Please write your solution to Problem 4 here:

```
def is_graduate_course(course):
```

_____ (Your SUID, e.g. 006892456, required on every page)

Use this page as additional space for Problem 4, if needed.

This page intentionally left blank.

Problem 5: Sum To Value (16 points)

Write a function **`sum_to_value`** that takes a 2-dimensional list of integers named **`sequences`** as a parameter, along with an integer value **`desired_sum`**. It is your job to add **at most 1** element to each "internal" list in **`sequences`** to ensure that once the function ends, each internal list sums to **`desired_sum`**. Then return **`sequences`** back to the caller.

For example, say your input list **`sequences`** is:

`[ [1, 3, 2], [5, 4, 3] ]`

and **`desired_sum`** is 10. The first internal list **`[1, 3, 2]`** sums to 6 so you append 4 to that list in order for it to sum to 10. The second internal list **`[5, 4, 3]`** sums to 12 so you append -2 to that list in order for it to sum to 10.

In the case that one of the internal lists already sums to **`desired_sum`**, leave that internal list unedited. Do not add any values to the internal list in this case.

Some notes on this problem:

- An internal list could be empty.
- The internal lists will only ever contain integers.
- You should modify the 2-dimensional list **`sequences`** in place, rather than populating and returning a new 2-dimensional list.
- Some of you may be familiar with the **`sum`** function for lists. You may *not* use this function for this problem.

Here are some sample outputs of this function, expressed as doctests:

```
>>> sum_to_value([[1, 3, 2], [5, 4, 3]], 10)
[[1, 3, 2, 4], [5, 4, 3, -2]

>>> sum_to_value([[1, -1], [3, 6], [-5]], 0)
[[1, -1], [3, 6, -9], [-5, 5]]

>>> sum_to_value([[5, 3, -8, 7, 2]], 15)
[[5, 3, -8, 7, 2, 6]]

>>> sum_to_value([[8, 0], []], 10)
[[8, 0, 2], [10]]
```

Please write your solution to Problem 5 here:

```
def sum_to_value(sequences, desired_sum):
```

_____ (Your SUID, e.g. 006892456, required on every page)

Use this page as additional space for Problem 5, if needed.

This page intentionally left blank.