CS106A Summer 2025

Practice Midterm

Border Karel	15
Largest and Second Largest	16
Collapse List	12
Longest Consonants	12
Make True Grid	18
	73

_____

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 header files are available to you. You are free to create helper functions unless the problem states otherwise. 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.

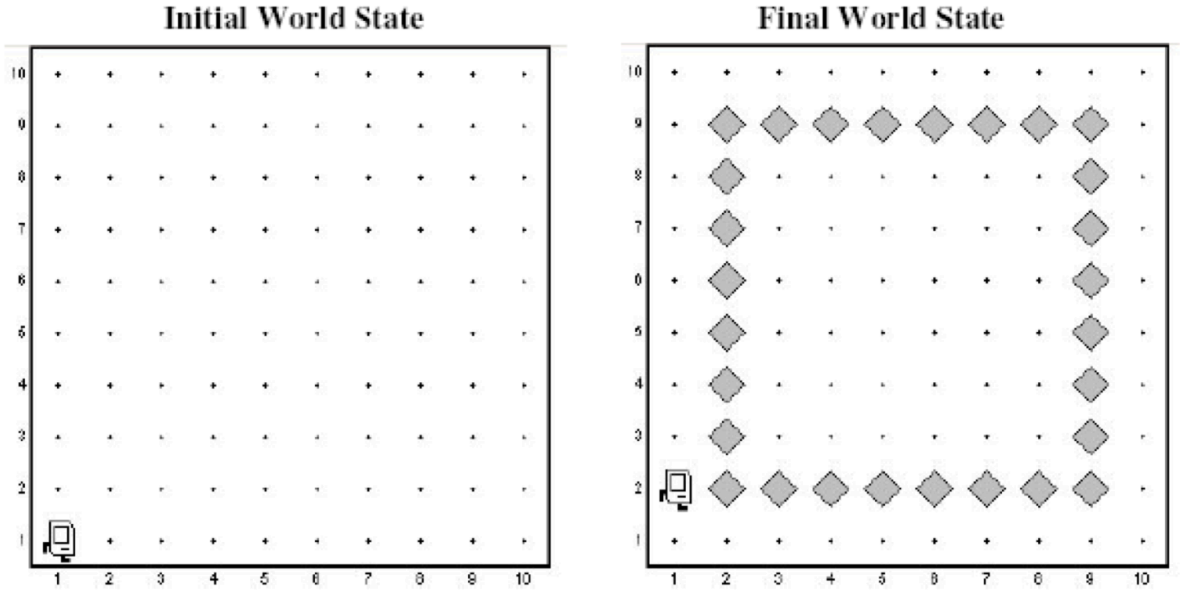
_____(signature)

I accept and acknowledge the honor code.

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

Problem 1: Border Karel (15 points)

We want to write a Karel program which will create an inside border around the world. Each location that is part of the border should have one (and only one) beeper on it and the border should be inset by one square from the outer walls of the world like this:



In solving this problem, you can count on the following facts about the world:

- You may assume that the world is at least 3x3 squares. The correct solution for a 3x3 square world is to place a single beeper in the center square.
- Karel starts off facing East at the corner of 1st Street and 1st Avenue with an infinite number of beepers in its beeper bag.
- We do not care about Karel's final location or orientation.
- You are limited to the instructions in the Karel course reader. Notably, you cannot use variables except as the index variable in for loops (such as the variable `i`) and you cannot refer to that variable otherwise.

_____ (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__”:
      run_karel_program()
```

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

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

Problem 2: Largest and Second Largest (16 points)

For this problem, write a program that asks the user to enter integers until they enter a 0 as a sentinel to indicate the end of the input list. Your program should then print out the largest and second-largest values from values the user entered. A sample run of the program might look like this (user input is the numbers after "Enter value:"):

```
Enter value: 7
Enter value: 42
Enter value: 18
Enter value: 9
Enter value: 35
Enter value: 0
The largest value is 42
The second largest is 35
```

To reduce the number of special cases, you may make the following assumptions:

- The user must enter at least two values before the sentinel.
- All values inputted by the user are positive integers.
- If the largest value appears more than once, that value should be listed as both the largest and second-largest value, as shown in the following sample run:

```
Enter value: 1
Enter value: 8
Enter value: 6
Enter value: 8
Enter value: 0
The largest value is 8
The second largest is 8
```

Please write your solution for Problem 2 here.

```
SENTINEL = 0    # the sentinel used to signal the end of the input
```

```
if __name__ == “__main__”:
    main()
```

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

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

Problem 3: Collapse List (12 points)

Write a function `collapse` that accepts a list of integers as a parameter and returns a new list containing the result of replacing each pair of integers with the sum of that pair. If the list stores an odd number of elements, the final element is not collapsed. You must not edit the input list.

Here are some sample calls to `collapse`, shown as doctests.

```
>>> collapse([7, 2, 8, 9, 4, 13, 7, 1, 9, 10])
[9, 17, 17, 8, 19]
>>> collapse([1, 2, 3, 4, 5])
[3, 7, 5]
```

Please write your solution to Problem 3 here:

```
def collapse(lst):
```

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

Use this page as additional space for Problem 3, if needed.

Problem 4: Longest Consonants (12 points)

Write a function `longest_consonants` that takes in a string `s` as a parameter and returns the longest number of consonants that appear in a row within that string. We'll consider consonants to be letters that are not A, E, I, O, or U. If `s` does not contain any consonants, you should return 0. There may also be non-alphabetic characters in `s`, which do not count as consonants.

Notes:

- To check if a character is a vowel, you can check if that character is in the string `"AEIOU"`.
- You may assume that the string `s` is all lowercase.

Here are a few sample calls to `longest_consonants`, shown as doctests.

```
>>> longest_consonants('rhythm')
6
>>> longest_consonants('what is the longest consonant here')
2
>>> longest_consonants('aaaaaa')
0
```

Please write your solution to Problem 4 here:

```
def longest_consonants(s):
```

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

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

Problem 5: Make True Grid (18 points)

When working with 2-dimensional lists in Python, it can often be easier to work with a "true" grid where every row in the 2-dimensional list has the same number of elements, as opposed to a "jagged" grid where each row may have a different number of elements.

Your job is to write a function called `make_true_grid(grid, row_size, filler)` that is passed a 2-dimensional list of integers (`grid`), the minimum row size for each row of the grid after the grid is made "true" (`row_size`), and an integer "filler" value (`filler`) that will be used to pad rows in the original grid to make them all the same size.

Your function should modify the grid passed in so that every row has the **same number** of elements. If the original grid passed in has any rows larger than `row_size`, then each row in the grid after your function is done should be the size of the longest row (row with the most elements) in the original grid passed in (i.e., you should guarantee that all the elements in the original grid show up in the final grid). If the length of all the rows in the original grid are less than or equal to `row_size`, then each row in the grid after your function is done should have the length `row_size`.

In order to make the length of each row in the grid the same size, you should add the `filler` value to the ends of those rows, as needed, to make them all the same size. There should be no extra rows added to the grid.

Here are some examples of calls to `make_true_grid`, and what the initial grid and resulting grid (after the function is called), should look like:

```
grid = [[2, 3], [5], [1, 2, 3, 4, 5]]
# After calling make_true_grid(grid, 3, 0), the grid should be:
[[2, 3, 0, 0, 0], [5, 0, 0, 0, 0], [1, 2, 3, 4, 5]]

grid = [[10, 20], [], [1]]
# After calling make_true_grid(grid, 4, 9), the grid should be:
[[10, 20, 9, 9], [9, 9, 9, 9], [1, 9, 9, 9]]

grid = [[1, 2], [3, 4], [5, 6]]
# After calling make_true_grid(grid, 1, 0), the grid should be:
[[1, 2], [3, 4], [5, 6]]
```

As a side note (although it will not explicitly impact your code), if an already true grid (all the rows are the same length and the the length is greater than or equal to `row_size`) is passed to your function, that grid should not be changed by your function (as shown in the last example above).

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

Please write your solution to Problem 5 here.

```
def make_true_grid(grid, row_size, filler):
```

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

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