CS 2: Introduction to Programming Methods — Winter 2026

Week02 DUET Worksheet

Demo

We will ask you to play a game of Evil Hangman in your terminal.

Understand

We will discuss the concept of nested data structures (a data structure inside another data structure) and how they were used in project02.

Experiment

The Problem

Pasadena has more restaurants per capita than New York City. With so many choices, it's hard to get a large group of friends to agree on which one to eat at for dinner.

Approval Voting

To decide where to go, every friends will vote on restaurants they approve of (i.e., "are okay going to"). Each person can approve of an unlimited number of choices. The restaurant with the highest number of approvals wins and is selected; ties are broken by alphabetical order.

Example

Inputs

- The **three** friends choosing where to eat are Aiden, Bailey, and Classic.
- The **four** available restaurants to vote for are Nine & Nine, BCD Tofu House, Main Chick, Zankou Chicken.
- Their approval votes are as follows:
  - Aiden approves of BCD Tofu House and Nine & Nine
  - Bailey approves of Nine & Nine and Zankou Chicken
  - Classic approves of BCD Tofu House, Main Chick, and Nine & Nine

Output

The winner is **Nine & Nine** (with 3 votes) since every other option has two or fewer votes.

Data Structures to the Rescue!

As we've discussed in class, a large part of solving algorithmic problems is **choosing the right data structure**. In this Experiment, you'll write an algorithm to solve this problem using two different data structures (a `List` and a `Map`). Both versions will be similar to the algorithms and data structures we developed for `IntBag` in the first lecture.

Option 1: Using a List

Design and be ready to describe how we could use a `List` as a backing data structure to solve the problem. Your algorithm will need to have two parts: (1) constructing the data structure and (2) retrieving the output from the data structure. You may not use any data structures other than a list, and you may **only** have a single list.

Option 2: Using a Map

Design and be ready to describe how we could use a `Map` as a backing data structure to solve the problem. Your algorithm will need to have two parts: (1) constructing the data structure and (2) retrieving the output from the data structure.

Theory

We define the following two variables to analyze how good each solution is:

- Let r be the number of restaurants being considered.
- Let f be the number of friends voting.

(a) Consider your first algorithm (option 1) which used a `List`:

- **(i)** In the worst case, how many items are in the list?
- **(ii)** In the worst case, how many times is each item in the list accessed?

(b) Consider your second algorithm (option 2) which used a `Map`:

- **(i)** In the worst case, how many items are in the map?
- **(ii)** In the worst case, how many times is each item in the map accessed?

(c) Use your answers to (a) and (b) to explain why option 2 is likely a better choice to solve this problem.