CSCM712 – Problem efficiency and solving software March 27th 2026
Cécilia Pradic & Deshan Sumanathilaka

Lab 9: hashing

For the lab this week, we expect you to put more code in the file `hashing.py` for Task 9.2 and `countdown.py` for Task 9.3. Task 9.1 is to be done in a file you create for youself. Please write tests for your functions before you sign off.

1. **Using hashtables**

   (a) Write a function

   ```
   def invertArray(arr):
   ```

   returning a hashtable `h` such that, for any string `s`, `h[s]` returns the first index `i` of `arr` such that `arr[i] == s`.

   (b) Write a function

   ```
   def countSort(arr):
   ```

   which sorts a list of size n containing at most K distinct values in time $\mathcal{O}(nK)$ on average using a dictionary.

2. **Implementing hashtables** Open the file `hashing.tex` and reminisce how things in there are supposed to work (ask questions during the lab if you don't get it). Complete the code for the methods

   ```
   def delitem(self, key):
   ```

   of the classes `chainingHashTable` and `probingHashtable`. Write some tests. For `probingHashtable`, your test should evidence the working of tombstone markers.

3. **Countdown!** *Countdown* is a British[1] TV game show that is still airing nowadays. In one of the games, the contestants are given a list of numbers, as well as a target number, and attempt to reach the target number by applying the basic arithmetic operations `+`, `*`, `-` and `/` using the numbers they are given. There are some natural restrictions in how they can use those numbers:

   - Negative or fractional numbers are not allowed at any stage of a computation. So for instance, $(3-5)+10$ and $5/2+3/2$ are never valid answers.
   - All numbers given to reach the target must be used at most once in the computation. For instance, if the list $5,5,4,3,2$ is given, $5+5-4$ and $3*2$ are valid answers to reach 6, but $5-4+3+4-2$ is not.

[1] Actually a French export; not sure where else TV shows with the same conceit air.

Write the body of the function

```
def countdown(pbm):
```

that returns an arithmetic expression which is a solution of the countdown problem for the list `pbm` if it exists, and `None` otherwise. Feel free to write your own auxiliary methods!

There are two things you will have to use:

- You may use the method

  ```
  def allsplits(xs):
  ```

  which is provided. It enumerates in a list all possible ways of splitting a tuple of elements `xs` in two subtuples `ys` and `zs` different from `()` that enumerate the same elements as `xs` and, say, in the same order in both tuples as in `xs`. For instance, `allSplits` applied to `(3,2,1,0)` returns

  ```
  [((3, 0), (2, 1)), ((3,), (2, 1, 0)), ((3, 1, 0), (2,)), ((3, 1), (2, 0))]
  ```

- You should use a global hash table `memoAllExpressions` to do dynamic programming to avoid many recomputations. This hash table will store the set of all arithemtic expression you may produce using a given set of numbers.
- (you should feel free to exploit the commutativity of addition and multiplication to cut down on the number of results you store in `memoAllExpressions` (i.e., it's probably not very useful to store two solutions that read off as the same result, such as `1 + 2` and `2 + 1`)