Data Structures

Hashing Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Handling all letters and digits

- **def** hash_string(str)
- Implement a hash function that converts a string to an integer, following the approach in the lecture
- Change the function to handle lowercase letters (a-z), uppercase letters (A-Z) and digits (0-9)

Problem #2: Folding for hashing

- There are several hashing techniques, including Binning and Mid-Square
  - Feel free to google them
- Let's implement a string folding technique as follows:
  - Compute the hash value for each block of 4 consecutive letters
  - Sum all blocks, returning their final hash
- For example, take the input "aabcdefgAxT334gfg"
  - The groups are: aabc, defg, AxT3, 34gf, g
  - Hash each one (using previous problem). Sum all hashes
- **def** hash_string_folding(str)

Problem #3: Rehashing

- We need to change our code to allow rehashing:
  - **def** __init__(self, table_size, limit_load_factor = 0.75):
    - This imposes a limit on the load factor. When it is exceeded, we need to rehash!
- **def** _rehash(self):
  - Rehash the table content to use **double** the size of the previous one

```
dct = OurDict(10, 0.5)

dct.add('Mostafa', 1)
dct.add('Ziad', 2)
dct.add('Ali', 3)
dct.add('Amal', 4)
dct.add('Hany', 5)
dct.add('Belal', 6)
dct.add('Safaa', 7)
dct.add('Safa', 8)
dct.add('Ashraf', 9)
dct.add('Morad', 10)
dct.add('John', 11)
```

```
Rehashing - new size will be: 20
Rehashing - new size will be: 40
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."