United International University
Department of Computer Science and Engineering
CSE-2218 Data Structures and Algorithms II Laboratory

Assignment Zero — Set: A — Summer 2025

Total Marks: **25** Time: Date:

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

Name: ID:

---

1. Welcome to the magical world of Hogwarts. For centuries magicians have lived here peacefully. Recently, however, they have faced a problem: some muggles are entering Hogwarts and are mistakenly treated as magicians.

   Every person in Hogwarts is identified by a positive integer. If the sum of the digits in even positions is odd, the person is considered a magician; otherwise, they are a muggle. (For example: 1324; Here 3+4 = 7, even number; therefore, it is a magician!)

   Write a recursive function (a magic spell that calls itself) that determines whether a given ID number belongs to a magician or a muggle.

Sample Input	Sample Output
1324	Magician
245	Muggle

   [You can assume that the input numbers will have at least 2 digits and the leftmost digit is in position 1.]

2. At *Big Bang High School*, **n** students await their exam results, but their teacher, Professor Sheldon Cooper, is too lazy to check the scripts. The principal demands results within 96 hours, so the professor invents a quirky plan: rank students by the "magical value" of their names — the sum of the ASCII values of all characters.

   Sorting rules:

   - Higher magical value comes first.
   - If tied, the student with the lower roll number comes first.

   Write a program (using a **custom comparator** along with the **STL** algorithm) to sort and display the students accordingly.

Sample Input	Sample Output
5 101 Alice 102 Bob 103 Charlie 104 Dave 105 Eve	103 Charlie 101 Alice 104 Dave 105 Eve 102 Bob

3. Deep inside the *Cave of Illusions*, a traveler is trapped by an ancient wizard. The only way to escape is to solve the wizard's puzzle: three enchanted crystal pillars stand in the center of the cave, with **n** magical discs of different sizes stacked on the first pillar.

   The discs are arranged so that the largest is at the bottom and the smallest is at the top. The wizard's rules are strict:

   - Only one disc may be moved at a time.
   - A disc may be placed only on top of a larger disc or on an empty pillar.
   - All n discs must be moved from the first pillar (source) to the third pillar (destination), using the second pillar (helper) if needed.

   Your task is to help the traveler escape by printing the sequence of moves that will transfer all the discs from the source pillar to the destination pillar, following the rules.

   Input: $1 \leq n \leq 20$
   Output: A sequence of moves in the format:

   ```
   Move disc X from pillar A to pillar B
   ```

   where X is the disc number (1 = smallest, n = largest), and A and B are pillar labels (e.g., 1, 2, 3).

Sample Input	**Sample Output**
3	Move disc 1 from pillar 1 to pillar 3 Move disc 2 from pillar 1 to pillar 2 Move disc 1 from pillar 3 to pillar 2 Move disc 3 from pillar 1 to pillar 3 Move disc 1 from pillar 2 to pillar 1 Move disc 2 from pillar 2 to pillar 3 Move disc 1 from pillar 1 to pillar 3

   Hints:

   (a) You might want to check the **link**.

   (b) Algorithm for Tower of Hanoi:

```
Hanoi(n, source, helper, destination):
    If n == 1:
        Print "Move disc 1 from pillar source to pillar destination"
        Return

    Hanoi(n - 1, source, destination, helper)
    // Move top n-1 discs from source to helper

    Print "Move disc n from pillar source to pillar destination"
    // Move the largest disc directly to destination

    Hanoi(n - 1, helper, source, destination)
    // Move the n-1 discs from helper to destination
```

4. Harry Potter is on a mission to destroy You-Know-Who's Horcruxes. The first Horcrux he encountered in the Chamber of Secrets is Tom Riddle's diary. The diary was with Ginny and it forced her to open the Chamber of Secrets. Harry wants to know the different people who had ever possessed the diary to make sure they are not under its influence.

   He has the names of n people who possessed the diary in order. You need to tell, for each person, if he/she possessed the diary at some point before or not.

   Formally, for a name s_i in the i-th line, output `"YES"` if there exists an index j such that $s_i = s_j$ and $j < i$, otherwise, output `"NO"`.

   Input: The first line of input contains an integer n ($1 \leq n \leq 100$) — the number of names in the list. Next n lines each contain a string s_i, consisting of lowercase English letters. The length of each string is between 1 and 100.

   Output: Output n lines, each containing either `"YES"` or `"NO"`, depending on whether this string was already present in the stream or not. You can print each letter in any case (upper or lower).

Sample Input	Sample Output
6 tom lucius ginny harry ginny harry	NO NO NO YES YES

5. A bracket is considered to be any one of the following characters: `( ), { }, [ ]`.

   Two brackets are considered to be a matched pair if an opening bracket (i.e., `(`, `[`, or {) occurs to the left of a closing bracket (i.e., `)`, `]`, or }) of the exact same type. There are three types of matched pairs of brackets: `[]`, {}, and `()`.

   A sequence of brackets is said to be **balanced** if:

   - It contains no unmatched brackets.
   - The subset of brackets enclosed within any matched pair of brackets is also balanced.

   Given n strings of brackets, determine whether each sequence is balanced. If a string is balanced, output `YES`, otherwise output `NO`.

   Input: The first line contains an integer n ($1 \leq n \leq 10^3$) — the number of strings. Each of the next n lines contains a string consisting only of the characters `( ), { }, [ ]`, with length between 1 and 10^3.

   Output: For each string, print `YES` if it is balanced, otherwise `NO`.

Sample Input	Sample Output
3 {[()]} {[(])} {{[[(())]]}}	YES NO YES

6. Write a recursive program to print all subsets of a set of n elements.

   Input: The first line contains an integer n ($1 \leq n \leq 20$) — the number of elements in the set. The next line contains n distinct integers, separated by spaces.

   Output: Print all possible subsets of the set, one per line. The order of the subsets does not matter. Each subset's elements should be separated by a space. The empty set should also be printed.

Sample Input	**Sample Output**
3 1 2 3	1 2 3 1 2 1 3 2 3 1 2 3