

# **CSE216**

# **Foundations of Computer Science**

**Instructor: Zhoulai Fu**

**State University of New York, Korea**

**C Lab + C Mock Final**

# Lab exercise 1:

## Implementing a Caesar Cipher in C

- In this lab exercise, you will be implementing a simple Caesar cipher in the C programming language. A Caesar cipher is a type of substitution cipher where each character in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on.

# Problem Statement

```
int main() {  
    char str[] = "KENNEDY";  
    caesarCipher(str);  
    printf("%s\n", str); // Should print "NHQQHGB"  
    return 0;  
}
```

- Write a C function **void caesarCipher(char\* str)** that performs a Caesar cipher on an input string. where each character in the plaintext is 'shifted' a certain number of places down the alphabet.
- The string will consist of capital letters only, and the cipher should shift each letter 3 places to the right in the alphabet, wrapping around to the beginning of the alphabet if necessary. For example, with a shift of 3, A would be replaced by D, B would become E, Z will be replaced by C, and so on.
- For example, the input string "KENNEDY" should produce the output "NHQQHGB".

# FYI

- **Character arrays in C:** In C, strings are typically represented as arrays of characters. For example, the string "HELLO" can be declared as **char str[] = "HELLO";**. Note that all strings in C are null-terminated, which means they end with a special character '\0'.
- **Character pointers in C (char\*):** A character pointer in C can also be used to represent a string. It can point to the first character of a string, and the string is assumed to continue until a null character is encountered. For example, **char\* str = "HELLO";**.
- **String manipulation in C:** C provides several functions for manipulating strings, such as **strcpy** for copying strings and **strlen** for finding the length of a string. However, in this exercise, you will be manipulating strings directly.

# Lab exercise 2:

## Sentence Title Case Verification in C

- Your task is to write a C function that checks whether a sentence is in 'Title Case'. In other words, the function should return true if each word in the sentence starts with a capital letter and continues with lowercase letters. Here are the specific requirements:
  - The function should take a single argument - a string, representing the sentence to check. This string consists only of letters and blank spaces.
  - The function should return a boolean value (in C, typically represented as an int with 0 for false and non-zero for true).
  - The function should return true if and only if each word in the sentence starts with a capital letter and continues with lowercase letters. Otherwise, it should return false.
- Write the function as described above. Test your function with several test sentences to ensure that it works correctly.

# FYI

- In C, strings are represented as arrays of characters. You can use array indexing to access individual characters in a string, similar to how you'd access elements in an array. For example, `sentence[0]` would give you the first character in the string `sentence`.
- C provides functions to manipulate and check characters. You might find the following functions from the `ctype.h` library useful:
  - **`isupper(int c)`** checks if the given character is uppercase.
  - **`islower(int c)`** checks if the given character is lowercase.
  - **`isspace(int c)`** checks if the given character is a whitespace character.
- Reminder: A string in C is null-terminated, meaning it ends with the special null character `'\0'`. You can use this fact to iterate through the string.