Department of Computer Science and Engineering,
PES University, Bangalore, India

Lecture Notes
Problem Solving With C
UE24CS151B

Lecture #5
Problem Solving using Strings Functions

By,
Prof. Sindhu R Pai,
Theory Anchor, Feb-May, 2025
Assistant Professor
Dept. of CSE, PESU

Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Center, PES University)
Prof. Nitin V Poojari (Dean, Internal Quality Assurance Cell, PES University)

Unit #: 3

Unit Name: Text Processing and User-Defined Types

Topic: Problem Solving using String Functions

Course objectives: The objective(s) of this course is to make students

- Acquire knowledge on how to solve relevant and logical problems using computing Machine.
- Map algorithmic solutions to relevant features of C programming language constructs.
- Gain knowledge about C constructs and its associated ecosystem.
- Appreciate and gain knowledge about the issues with C Standards and it's respective behaviours.

Course outcomes: At the end of the course, the student will be able to:

- Understand and Apply algorithmic solutions to counting problems using appropriate C Constructs.
- Understand, Analyze and Apply sorting and Searching techniques.
- Understand, Analyze and Apply text processing and string manipulation methods using Arrays, Pointers and functions.
- Understand user defined type creation and implement the same using C structures, unions and other ways by reading and storing the data in secondary systems which are portable.

Sindhu R Pai

Theory Anchor, Feb - May, 2025

Dept. of CSE,

PES University

Solve the below programs using String functions.

Link for Solution:
https://drive.google.com/file/d/1yqkXiZJinZpyOcSjuh24_HrSJ1UXKNq1/view?usp=drive_link

Level-1: Banana

1. In a college's student portal system, security is a top priority. The admin, Arjun, is developing a login feature that ensures **the password should be at least 8 characters long**. Arjun wants to alert users if their passwords are too short so that their accounts stay safe from easy guessing or brute-force attacks. Help him **implement this password length checker.**

2. Meena, a lab assistant in the computer science department, is developing a simple login system for the lab's admin panel. Only authorized users are allowed to change system configurations or access student records. To begin with, she **hard codes the registered username as "student".** Whenever a user tries to log in, **the system should verify if the typed username matches the registered one exactly. If it does, access is granted. Otherwise, the system denies entry to ensure unauthorized users can't proceed.**
   Input: student
   Expected Output: Login successful
   Input: STUDENT
   Expected Output: Incorrect username

3. At a tech event, organizers greet each participant with a personalized badge. When someone arrives, their **name is added right next to the word "Welcome" to create a friendly message**. This small touch makes everyone feel recognized and valued as soon as they enter.
   Input: sindhu r pai
   Expected Output: Welcome sindhu r pai
   Input: 123
   Expected Output: Welcome 123

4. Kiran is building a smart feedback system for his college's virtual assistant chatbot. Since the chatbot processes text in real time, any **feedback longer than 100 characters causes delays in analysis**. To ensure smooth interaction, Kiran adds **a message length checker.** If a student's **feedback exceeds the character limit, the system warns them to shorten it. Otherwise, it proceeds with processing.**

5. Imagine you're part of the college event committee, and every week you ask students about their favorite day. If someone says **"Friday," it means it's party time!** Your task is to write a program that checks if the entered day is "Friday" and **prints a special message for that day.**
   Example #1:
   Input : Friday
   Expected output : Party Time!

Example #2:
Input : friDay
Expected output : Party Time!
Example #3:
Input : Monday
Expected output : Not Friday yet.

6. Anvi's college friends love texting each other long messages during study breaks. But Anvi only has a limited SMS pack where each space counts as a character, and she wants to save every character possible. Help Anvi by writing **the implementation of remove_space** function looking at the main program so that **message without space is stored somewhere and sent for further processing.**
Main Program:
```
char str[100], result[100];
printf("Enter a message : ");
scanf(" %[^\n]", str);  // read full line including spaces
remove_spaces(str, result);
printf("%s\n", result);
```
Example #1:
Input : Hello World health day
Expected output : HelloWorldhealthday

7. In your coding bootcamp, there's a fun mini-game where **students see their names written backward on a scoreboard**. It's part of a challenge to get students excited about string manipulation. You're assigned to build this small but cool feature! The task is simple: **Reverse any string the user enters and display it instantly.**
Samples:
Enter a word: sindhu r pai
Reversed: iap r uhdnis

8. In a startup building a smart email app, you're asked to help validate user input. The team wants a simple check to see if the entered text looks like an email. Your job is to **find out whether the @ symbol is present in the string.** This small check helps users avoid silly mistakes and keeps the app smart and clean. Can you help the system spot valid email patterns?
Example #1:
Input : thaksh@gmail.com
Expected output : Valid Email
Example #2:
Input : studentemailgmail.com
Expected output : Invalid Email
Example #3:
Input : @
Expected output : Valid Email

9. In your college's online exam system, there's a strict rule: students are not allowed to use certain special characters in their answers. For example, the character '$' is forbidden because it might be used to insert unauthorized formulas. Your task is **to help build a small tool that scans a student's answer and checks for the presence of this forbidden character**. If the answer contains '$', alert the system with "Forbidden character found". Otherwise, confirm that the answer is clean.
   Example #1:
   Input: The cost is $50 per item
   Expected output: Forbidden character found
   Example #2:
   Input: The cost is Rs.50 per item
   Expected output: Answer is clean

10. In your college tech fest, you're volunteering to build a small file submission tool for coding competitions. Since participants **must only upload .txt files containing their code or answers**, you need a filter that checks whether the uploaded file is in the correct format. This simple checker ensures everyone follows the rules, avoids viruses or unwanted formats, and keeps the system clean and secure. As a student developer, you're learning how real-world systems validate file types before processing them.
    Example #1:
    Input: Hello.txt
    Expected output: Valid file type
    Example #2:
    Input: Hello txt
    Expected output: Invalid file type
    Example #3:
    Input: txt .txt hello
    Expected output: Invalid file type
    Example #4:
    Input: txt hello .txt
    Expected output: Valid file type

Level-2: Orange

11. In a college, every student needs a unique username for the online portal. The **username is created using the student's first and last name, all in lowercase, separated by an underscore**. You're assigned to automate this task for the student database team. This will help speed up student registrations and make the process error-free. Be ready to help this college with simple automation.
    Example #1:
    Input1 :John
    Input2: Doe
    Expected output: john_doe
    Example #2:
    Input1 :Sindhu R
    Input2: Pai
    Expected output: sindhu r_pai

12. In your college coding club, your team is building a "**Word Wizard**" app that detects cool words. Your job is to design a **tool that checks if a word reads the same backward as forward** — like ***madam***, ***level*****, or *racecar*.** It's a small challenge, but super fun and helpful for real-world apps like spell-checkers or secret word puzzles!
    Example #1:
    Input: racecar
    Expected output: Palindrome
    Example #2:
    Input: racecaR
    Expected output: Palindrome

13. If you are part of the editorial board for the student magazine and you are building a text cleanup tool to help **filter out repeated letters from article titles**. This helps to visualize unique character sets, which can be useful for artistic covers or simple data compression. Your team relies on your tool to clean the input and **highlight only the first occurrence of every letter in capital case.**
    Example #1:
    Input : programming
    Expected output: PROGAMIN
    Example #2:
    Input : racECAr
    Expected output: RACE

14. In a college spy club, students love creating secret messages. One day, they receive a strange coded note and must analyze **how many times each character appears** to crack the pattern. They know the **key to decoding is in the frequency of each letter.** Your job is to write a program that helps them break the code by counting how often each **alphabetic character appears in a message**. **Ignore numbers and symbols — only letters matter to spies!**
    Example #1:
    Input : Hello WorLd!
    Expected output :
    H: 1
    e: 1
    l: 2
    o: 2
    W: 1
    r: 1
    L: 1
    d: 1
    Example #2:
    Input : 123sshhh123
    Expected output :
    s: 2
    h: 3

15. You're creating a simple account system for a school coding club website. The system **generates usernames in lower case by taking the student's first name, adding an underscore, and then the first two letters of their last name.** However, **usernames must be at most 10 characters total.** Help the developer to get this done.
Example #1:
Input1: Maria
Input2: Johnson
Expected output: maria_jo
Example #2:
Input1: sindhurpai
Input2: sagara
Expected output: Username too long

Level-3: Jackfruit

16. In your C programming class, you've been given a fun mini-project called the "Mirror Code." The task is inspired by **lapindromes—strings that, when split into two halves, have the same characters with the same frequency. If the string length is odd, you simply ignore the middle character and compare the rest.** Think of it like **building a security system that only accepts codes balanced like a mirror**. It's a great way to apply your knowledge and understand how simple checks like this are used in password validation, encryption, and data integrity verification in real applications.
Example #1:
Input: racecare
Expected output : YES
Example #2:
Input: raceCARE
Expected output : NO

17. Aarya is developing a smart programming assistant that helps students **find useful code snippets in a large collection of examples**. When a student enters a piece of code or keyword (like "scanf"), the assistant needs to check if that keyword exists inside any stored code sample. To do this, **Aarya writes a program that checks if one string (S2) is a substring of another string (S1).** This helps the assistant quickly identify relevant code and save students time during coding or debugging sessions.
Example #1:
Input: Hello ell
Expected output : YES.
Example #2:
Input: Hello hello
Expected output : NO.
Example #3:
Input: sindhu ind
Expected output : YES.

18. Thaksh is developing a **global travel booking app** where users from different countries enter dates in different formats—some use DD/MM/YYYY, others use MM/DD/YYYY. To avoid confusion while processing bookings, Thaksh **writes a tool that checks the input and figures out the correct format by analyzing the day and month values. If both values are 12 or less, the format could be either, so the app labels it as "BOTH"**. This clever trick helps the app interpret dates correctly and avoid travel mix-ups!
    Example #1:
    Input : 21/05/2001
    Expected output : DD/MM/YYYY
    Example #2:
    Input : 10/15/2069
    Expected output : MM/DD/YYYY
    Example #3:
    Input : 05/11/1999
    Expected output : BOTH
    Example #4:
    Input : 32/32/1999
    Expected output : INVALID Format

19. As part of a cybersecurity project, Divya, a computer science student, is building a tool to monitor web traffic. Her goal is **to extract just the domain name from any given URL to check whether the user is accessing a trusted or suspicious site**. Since URLs come in different formats like https://, with or without www., and with paths like /login, she writes a program to intelligently extract just the domain name, no matter how the URL is formatted. This helps her create a clean list of websites students access most frequently.
    Samples:
    Input: https://www.example.com/page
    Expected Output: "Domain: example.com"

20. You've just joined the organizing committee of your college's TechFest. After the event, you're responsible for generating participation certificates for hundreds of attendees. But the names and topics submitted during registration are all randomly cased: some in lowercase, some screaming in uppercase, and some just weird. Your task is to **automatically fix the formatting before printing the certificates:** Each word should begin with a capital letter. BUT, keep acronyms like HTML, PYTHON as they are — all uppercase — because they are official and important.
    Example #1:
    Input : hello world
    Expected output : Hello World
    Example #2:
    Input : programming in PYTHON
    Expected output : Programming In PYTHON
    Hint :Usestrtok() to split the sentence into words using spaces. Then modify each word to title case unless it's already in all uppercase (acronym).

Happy Coding using String functions!