CSE 108 – Computer Programming Lab.

Lab 9

Sting Manupulation and Recursion

09/05/2025

Part 1(30pts). Implement a recursive function in C that prints all the digits of a non-negative integer from left to right, one by one, separated by spaces.

Requirements:

- You must write a recursive function that prints the digits of a number.
- The function must not use any loops (`for`, `while`, `do-while` are not allowed).
- You are not allowed to use arrays or strings to convert the number.
- Only standard C libraries (like `stdio.h`) may be used.
- The function must handle all non-negative integers correctly (e.g. 0, single-digit, multi-digit numbers).
- The digits must be printed **from left to right**, each separated by a space.
- You must write a `main()` function to test your recursive function with sample inputs.
- Example: int number = 12345; print_digits(number); // Output: 1 2 3 4 5

Part 2 (40 pts). Implement a recursive function in C that converts a binary number (entered by the user) to its decimal equivalent using recursion.

Requirements:

- Implement the function: int binaryToDecimal(int n, int power);
- This function takes two arguments:
- `n`: The binary number to be converted (integer input).
- `power`: The current power of 2 for the corresponding bit (starting from 0).
- The function should use recursion to compute the decimal equivalent.
- The program must: Prompt the user to enter a binary number (integer input).
- Ensure the program works for valid binary numbers and returns the correct result.
- The program should work for valid binary numbers, ensuring the binary input starts with 1.
- You are not allowed to use the math.h library. You must implement the pow function recursively yourself.

Function Specification:

- **Input:**
  - An integer representing a binary number (e.g., `1011`).
- **Output:**
  - The decimal equivalent of the binary number (e.g., `11` for `1011`).

Part 3(30 pts). Pattern Searching. Given **text** string with length **n** and a **pattern** with length **m,** the task is to prints all occurrences of **pattern** in **text.**

Note: You may assume that n > m, and maximum length will be 100.

Examples:

Input: *text = "THIS IS A TEST TEXT", pattern = "TEST"*
Output: *Pattern found at index 10*

Input: *text = "AABAACAADAABAABA", pattern = "AABA"*
Output: *Pattern found at index 0, Pattern found at index 9, Pattern found at index 12*