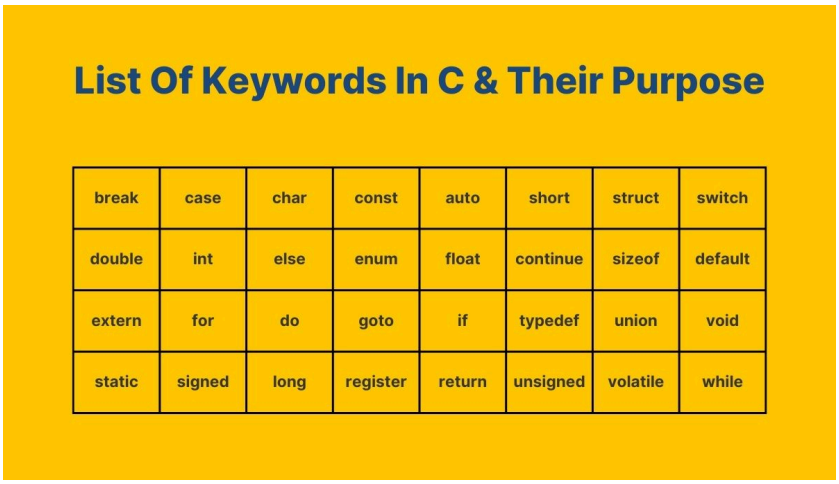
Problem Solving With C

Part 1: Fundamentals of C (Sessions 1-8)

This section covers the essential building blocks of the C language.

Session 1: Introduction to C & Basic Data Types

- **Topics:** Features and structure of a C program, the compilation process, fundamental data types (**int**, **float**, **char**, **double**), and variables.
- **Activities:** Write and compile a "Hello, World!" program and a program that uses different data types.
    1. Link : https://www.geeksforgeeks.org/c/c-hello-world-program/
    2. Link : Program Vs Process https://www.geeksforgeeks.org/operating-systems/difference-between-program-and-process/
- Link : Keywords :



Session 2: Operators & I/O

- **Topics: Arithmetic**, **relational**, **logical**, **bitwise**, **assignment**, and **ternary** operators. Standard input/output functions **printf()** and **scanf()**.
    1. Sum of Two Integers
    2. Subtract the Product and Sum of Digits of an Integer
    3. Add Digits
    4. Number of 1 Bits
    5. Counting Bits
    6. Divide Two Integers
    7. Minimum Bit Flips to Convert Number

Activities: Create a simple calculator program that takes user input and performs a calculation.

Session 3: Control Statements

- **Topics:** The **if**, **if-else**, and **switch-case** statements.
    1. **Power of Two**: https://leetcode.com/problems/power-of-two/
    2. **Two Sum:**https://leetcode.com/problems/two-sum/

- **Activities:** Write a program that uses a `switch` statement to handle a menu of options.

Session 4: Loops

- **Topics:** The **for**, **while**, and **do-while** loops.
- **Activities:** Use loops to print patterns or calculate a sum.

Solve using Iteration :

1. Print 1 to N Using Loops
2. **Palindrome Number**: https://leetcode.com/problems/palindrome-number/
3. **Single Number**: https://leetcode.com/problems/single-number/
4. **Valid Anagram**: https://leetcode.com/problems/valid-anagram/
5. Reverse Bits
6. Missing Number
7. Find the Difference
8. Fibonacci Number
9. Reverse Integer

Session 5: Functions & Recursion

- **Topics:** Function declaration, definition, and calls. **Pass by value** and **pass by reference**. The concept of **recursion**.

Solve using recursion

1. Print 1 to N Using recursion
2. Fibonacci Number
3. Reverse Integer
4. Reverse String

- **Activities:** Write a function to calculate a factorial using recursion.

Session 6: Arrays

- **Topics:** 1D and 2D arrays, initialization, access, and passing arrays to functions.
    1. Sequential Search in Array
    2. Binary Search Solve using Recursion and Iteration Both
    3. Remove Element
    4. Remove Duplicates from Sorted Array
    5. Plus One
    6. Single Number
    7. Contains Duplicate
    8. Rotate Array
    9. Intersection of Two Arrays
    10. Search in Rotated Sorted Array

11. Search in Rotated Sorted Array II

- **Activities:** Implement a simple program to find the largest element in an array.

Session 7: Strings

- **Topics:** Strings as character arrays. Built-in functions: **strlen**, **strcpy**, **strcmp**, and **strcat**.
    1. Reverse String
    2. Remove Outermost Parentheses
    3. Largest Odd Number in String
    4. Length of Last Word
    5. **Length of String : https://www.geeksforgeeks.org/problems/length-of-string/1**
    6. Count the Number of Consistent Strings
    7. Longest Common Prefix
    8. Rotate String

- **Activities:** Create a program that copies one string to another without using strcpy.

Session 8: Pointers

- **Topics:** Pointer declaration, dereferencing, and pointer arithmetic. Using pointers with arrays and strings.

Here is a list of the main pointer types in C:

- **Data Pointers**: Point to a variable of a specific data type (int *, char *, float *).
- **Null Pointer**: A pointer that points to no valid memory location. Its value is NULL.
- **Void Pointer**: A generic pointer that can point to any data type. It must be cast before dereferencing.
- **Wild Pointer**: A pointer that has been declared but not initialized, holding a random, invalid memory address.
- **Function Pointer**: A pointer that holds the memory address of a function.
- **Dangling Pointer**: A pointer that points to a memory location that has been deallocated.
- **Constant Pointers**: Pointers used to enforce immutability. This category includes:
    - **Pointer to a Constant**: The data is constant, but the pointer can be reassigned.
    - **Constant Pointer**: The pointer's address is constant, but the data can be modified.
    - **Constant Pointer to a Constant**: Both the pointer's address and the data are constant.

- **Activities:** Use pointers to swap the values of two variables.

Part 2: Advanced C Concepts (Sessions 9-15)

This section covers more complex topics essential for building robust C programs.

Session 9: Pointers & Functions

- **Topics:** Using pointers to pass arrays and strings to functions. **Function pointers**.
- **Activities:** Write a function that takes an array and its size as arguments and finds the average of its elements.

Session 10: Structures & Unions

A. **Topics:** Definition of **structures** and **unions**, accessing members, and **nested structures**.

B. **Activities:** Define a struct to represent a book and create an array of books.

Session 11: Pointers to Structures

- **Topics:** Declaring and using **pointers to structures**.
- **Activities:** Pass a structure to a function using a pointer to modify its members.

1. Reverse Linked List
2. Add Two Numbers
3. Middle of the Linked List
4. Linked List Cycle
5. Merge Two Sorted Lists
6. **Find Length of Linked List:** https://www.geeksforgeeks.org/problems/count-nodes-of-linked-list/1
7. **Search in Linked List :** https://www.geeksforgeeks.org/problems/search-in-linked-list-1664434326/1
8. **Kth from End of Linked List :** https://www.geeksforgeeks.org/problems/nth-node-from-end-of-linked-list/1

Session 12: Dynamic Memory Allocation (Part 1)

- **Topics:** The functions **malloc** and **calloc**.
- **Activities:** Create a dynamic array using malloc.

```
Struct {
Int id;
Char name [40;]
Float marks ;
}
```

a. Write a function to create a linked list Student Structure provided.
b. Write a function to delete the node at a given position and return the value of the deleted element.
c. Write a function which merges the two lists.

d. Write a function which merge two sorted list.
e. Create a Dynamic stack.

Session 13: Dynamic Memory Allocation (Part 2)

- **Topics:** The functions are realloc and **free**.
- **Activities:** Resize a dynamically allocated array using realloc.

Session 14: File Handling

- **Topics:** Opening, closing, reading, and writing files using **fopen**, **fclose**, **fscanf**, and **fprintf**.
- **Activities:** Write a program that saves user data to a text file.

Session 15: Preprocessor Directives

- **Topics:** The preprocessor and its directives: **#define**, **#include**, **#ifdef**, and **macros**.
- **Activities:** Create a simple header file and use #define to create a constant.