Practice (6-12-2025)

1. Calculate and display the factorial of a number `N`. Take `N` as an input from the user.

```
#include <stdio.h>

long long fact(int n);

int main() {
    int N;
    printf("Enter a number: ");
    scanf("%d", &N);
    printf("The Factorial of %d: ", N);
    printf("%lld", fact(N));
    return 0;
}

long long fact(int n) {
    if (n==0) {
        return 1;
    } else {
        return n * fact(n-1); // Recursion
    }
}
```

2. Calculate and print the sum of `N` natural numbers. Take `N` as an input from the user.

```
#include <stdio.h>

long long sumN(int n);

int main() {
    int N;
    printf("Enter a number: ");
    scanf("%d", &N);

    if (N!=0){
        printf("Sum of %d Natural numbers: ", N);
        printf("%lld", sumN(N));
    }

    return 0;
}

long long sumN(int n) {
    long long sum = 0;
    for (int i = 1; i<=n; i++) {
        sum += i;
    }
    return sum;
}
```

3. Calculate and display the `n`th term of Fibonacci Series.

```
#include <stdio.h>

int findFiboN(int N);

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    int result = findFiboN(n);
    if (n >= 0)
        printf("The %dth term of the Fibonacci Sequence: %d\n", n, result);
    return 0;
}

int findFiboN(int N) {
    int N1 = 0, N2 = 1, Nth = 0;

    if (N < 0) {
        printf("Enter a positive value\n");
        return -1;
    }

    if (N == 0) {
        return 0;
    }

    if (N == 1) {
        return 1;
    }

    for (int i = 2; i <= N; i++) {
        Nth = N1 + N2;
        N1 = N2;
        N2 = Nth;
    }

    return Nth;
}
```

4. Create a function which takes `N` as an input (`long long int`) and check whether it is divisible by `k` .

   Constraint :

   - `k` must be negative.
   - `k` is NOT EQUAL TO 0.

```
#include <stdio.h>
#include <stdbool.h>

bool checkDiv(long long N, long long k);

int main() {

    long long N, k;
    printf("Enter the value for N: ");
    scanf("%lld", &N);

    printf("Enter the value for k: ");
    scanf("%lld", &k);

    if (k>=0) {
        printf("Divisor (k) enterred must be negative.");
        return 0;
    } else {
        if (checkDiv(N,k)) {
            printf("The number %lld is divisible by %lld", N, k);
        } else {
            printf("The number %lld is not divisible by %lld", N, k);

        }
    }
    return 0;
}

bool checkDiv(long long N, long long k) {
    return (N%k==0);
}
```

5. a) Create a program to reverse a collection of strings in a 2-D array.

```
#include <stdio.h>

void reversePrint(char s[][20], int n) {
    printf("\n");
    for (int i = n - 1; i >= 0; i--) {
        printf("%s\n", s[i]);
    }
}

int main() {
    char str[10][20];
    int n = 3;  // number of strings - Name, Dept, College

    for (int i = 0; i < n; i++) {
        scanf("%s", str[i]);
    }

    reversePrint(str, n);

    return 0;
}
```

5. b) Order is the same but the elements are reversed.

```
#include <stdio.h>
#include <string.h>

// Function to reverse a single string
void reverseString(char str[]) {
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++) {
        char temp = str[i];
        str[i] = str[len - i - 1];
        str[len - i - 1] = temp;
    }
}

// Function to reverse all strings in a 2D array
void reverseAllStrings(char arr[][20], int n) {
    printf("\n");
    for (int i = 0; i < n; i++) {
        reverseString(arr[i]);
        printf("%s\n", arr[i]);
    }
}

int main() {
    char str[10][20];
    int n = 3;

    for (int i = 0; i < n; i++) {
        scanf("%s", str[i]);
    }

    reverseAllStrings(str, n);

    return 0;
}
```

6. You are given a string `s` and you have to determine whether it is Valid mobile number or not. Mobile Number is valid only if it is of length 10, consists of numeric values and it shouldn't have prefix zeroes.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

// Function to check if a string is a valid mobile number
int isValidMobile(char s[]) {
    int len = strlen(s);
    // Check length
    if (len != 10) {
        return 0;
    }

    // Check if first character is 0
    if (s[0] == '0') {
        return 0;
    }

    // Check if all characters are digits
    for (int i = 0; i < len; i++) {
        if (!isdigit(s[i])) {
            return 0;
        }
    }

    return 1;  // Valid number
}

int main() {
    int T;
    char s[20]; // max length 10, but safe to keep larger

    scanf("%d", &T);
    for (int i = 0; i < T; i++) {
        scanf("%s", s);
        if (isValidMobile(s)) {
            printf("YES\n");
        } else {
            printf("NO\n");
        }
    }
    return 0;
}
```

7. Using the Function Argument passing method, Find the average value of `n` values.

   Constraint:

   - `n` is of integer data type.

   1. Using `for` loop :

      ```
      #include <stdio.h>

      float findAvg(int N);

      int main() {
          int n;
          printf("Enter a number: ");
          scanf("%d", &n);

          printf("The Average of %d values: %.2f", n, findAvg(n));
          return 0;
      }

      float findAvg(int N) {
          float sum = 0;
          for (int i = 0; i<=N; i++) {
              sum += i * 10; // i-th multiple of 10
          }
          return sum/N;
      }
      ```

   2. Using `while` loop :

      ```
      #include <stdio.h>
      ```

```
float findAvg(int arr[], int N);

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    int numbers[n+1]; // 1-D array to store multiples of 10 up to N

    // Fill the array with multiples of 10 from 0 to N
    int i = 0;
    while (i <= n) {
        numbers[i] = i * 10;
        i++;
    }

    printf("The Average of %d values: %.2f\n", n, findAvg(numbers, n));
    return 0;
}

float findAvg(int arr[], int N) {
    float sum = 0;
    int i = 0;
    while (i <= N) {
        sum += arr[i];
        i++;
    }
    return sum / N;
}
```

8. Count the unique element of the array wherein the element value lies between 0 and 1001

```
#include <stdio.h>
```

```
#define MAX_VALUE 1001 // Constat

// Function to count unique elements
int countUnique(int arr[], int n) {
    int freq[MAX_VALUE + 1] = {0}; // frequency array

    for (int i = 0; i < n; i++) {
        freq[arr[i]]++;
    }

    int uniqueCount = 0;
    for (int i = 0; i <= MAX_VALUE; i++) {
        if (freq[i] == 1) {
            uniqueCount++;
        }
    }

    return uniqueCount;
}

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements (0 to 1001):\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int uniqueCount = countUnique(arr, n);
    printf("Number of unique elements: %d\n", uniqueCount);
    return 0;
}
```

9. Count the number of word in a sentence.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

// Function to count words in a sentence
int countWords(char str[]) {
    int count = 0;
    int i = 0;
    int inWord = 0; // flag to track if we are inside a word

    while (str[i] != '\0') {
        if (!isspace(str[i])) { // not a space
            if (inWord == 0) {
                inWord = 1; // start of a new word
                count++;
            }
        } else {
            inWord = 0; // we hit a space
        }
        i++;
    }

    return count;
}

int main() {
    char sentence[1000];

    printf("Enter a sentence:\n");
    fgets(sentence, sizeof(sentence), stdin);
    // Remove the newline character if present
    sentence[strcspn(sentence, "\n")] = '\0';

    int words = countWords(sentence);
    printf("Number of words: %d\n", words);

    return 0;
}
```

10. Write a C program to re-arrange given two strings and check if they are anagram of each other. Order of the word doesn't matter. Here the length should be equal.
    For example, “act” and “cat” are anagrams.

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

// Function to check if two strings are anagrams
bool areAnagrams(char str1[], char str2[]) {
    int count[256] = {0}; // ASCII character count array
    int i;

    // If lengths are not equal, not anagrams
    if (strlen(str1) != strlen(str2)) {
        return false;
    }

    // Count characters in str1
    for (i = 0; str1[i] != '\0'; i++) {
        count[(int)str1[i]]++;
    }

    // Subtract count using str2
    for (i = 0; str2[i] != '\0'; i++) {
        count[(int)str2[i]]--;
    }

    // If all counts are 0, they are anagrams
    for (i = 0; i < 256; i++) {
        if (count[i] != 0) {
            return false;
        }
    }

    return true;
}

int main() {
    char str1[100], str2[100];

    printf("Enter first string: ");
    scanf("%s", str1);
    printf("Enter second string: ");
    scanf("%s", str2);

    if (areAnagrams(str1, str2)) {
        printf("\"%s\" and \"%s\" are anagrams.\n", str1, str2);
    } else {
        printf("\"%s\" and \"%s\" are NOT anagrams.\n", str1, str2);
    }

    return 0;
}
```