`1. Election Vote Count (Using Arrays)`

`Problem Statement:`

A city conducts an election with `5 candidates`.

Citizens cast their votes (numbers `1 to 5` representing candidates).

Write a program to:

1. Read `N` votes into an array.

2. Count the total votes for each candidate.

3. Determine the `winner` (candidate with the highest votes).

```c
#include <stdio.h>



void countVotes(int votes[], int n, int candidateVotes[]) {

    for (int i = 0; i < 5; i++) {

        candidateVotes[i] = 0;

    }



    for (int i = 0; i < n; i++) {

        if (votes[i] >= 1 && votes[i] <= 5) {



            candidateVotes[votes[i] - 1]++;

        } else {

            printf("Invalid vote: %d\n", votes[i]);

        }

    }

}
```

```c
int findWinner(int candidateVotes[]) {
    int winner = 0;
    for (int i = 1; i < 5; i++) {
        if (candidateVotes[i] > candidateVotes[winner]) {
            winner = i;
        }
    }
    return winner;
}

int main() {
    int n;

    printf("Enter number of votes: ");
    scanf("%d", &n);

    int votes[n];
    printf("Enter votes (1-5): ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &votes[i]);
    }
```

```
    int candidateVotes[5];

    countVotes(votes, n, candidateVotes);



    for (int i = 0; i < 5; i++) {

        printf("Candidate %d: %d votes\n", i + 1, candidateVotes[i]);

    }



    int winner = findWinner(candidateVotes);

    printf("Winner: Candidate %d\n", winner + 1);



    return 0;

}
```

Enter number of votes: 15

Enter votes (1-5): 5

5

5

5

1

1

2

3

4

1

5

3

4

2

1

Candidate 1: 4 votes

Candidate 2: 2 votes

Candidate 3: 2 votes

Candidate 4: 2 votes

Candidate 5: 5 votes

Winner: Candidate 5

```
int main() {
    short int ages[] = {45, 42, 14, 20, 70};
    displayAges(ages, 5, 0);
    return 0;
}
```

for above driver code,

complete the definition of function displayAges

which will do forward traversal of ages array

using recursion.

And include all the libraries required.

```
void displayAges( int ages[], int size, int index) {
    if (index < size) {
```

```
        printf("%d ", ages[index]); // Print the current age

        displayAges(ages, size, index + 1); // Recursive call for the next
element

    }

}



int main() {

    int ages[] = {45, 42, 14, 20, 70};

    displayAges(ages, 5, 0);

    return 0;

}
```

`1. Student Marks Sorting (Using 2D Arrays)`

`Problem Statement:`

A college stores student marks in a `2D array` where:

- `Rows represent students`

- `Columns represent subjects`

1. Read marks for `N students` across `M subjects`.

2. Calculate and display each student's `total marks`.

3. Sort students in `descending order of total marks`.

```
#include <stdio.h>

#include <stdio.h>



#define MAX_SUBJECTS 10
```

```
void calculateTotalMarks(int marks[][MAX_SUBJECTS], int n, int m, int
totalMarks[]) {
    for (int i = 0; i < n; i++) {
        totalMarks[i] = 0;
        for (int j = 0; j < m; j++) {
            totalMarks[i] += marks[i][j];
        }
    }
}

void sortStudents(int totalMarks[], int n, int studentIds[]) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (totalMarks[j] < totalMarks[j + 1]) {
                // Swap total marks
                int temp = totalMarks[j];
                totalMarks[j] = totalMarks[j + 1];
                totalMarks[j + 1] = temp;

                // Swap student IDs
                temp = studentIds[j];
                studentIds[j] = studentIds[j + 1];
                studentIds[j + 1] = temp;
```

```
            }

        }

    }

}



int main() {

    int n, m;

    printf("Enter number of students: ");

    scanf("%d", &n);

    printf("Enter number of subjects: ");

    scanf("%d", &m);



    int marks[n][MAX_SUBJECTS];

    int totalMarks[n];

    int studentIds[n];




    for (int i = 0; i < n; i++) {

        printf("Student %d: ", i + 1);

        studentIds[i] = i + 1;

        for (int j = 0; j < m; j++) {

            scanf("%d", &marks[i][j]);

        }
```

```
    }



    calculateTotalMarks(marks, n, m, totalMarks);



    sortStudents(totalMarks, n, studentIds);



    printf("\nSorted Students by Total Marks:\n");
    for (int i = 0; i < n; i++) {
        printf("Student %d: Total Marks = %d\n", studentIds[i],
totalMarks[i]);
    }


    return 0;
}
```

Enter number of students: 5

Enter number of subjects: 5

Student 1: 78

90

78

55

46

Student 2: 99

09

34

76

36

Student 3: 45

33

11

78

90

Student 4: 7

54

32

56

09

Student 5: 98

68

68

57

90

Sorted Students by Total Marks:

Student 5: Total Marks = 381

Student 1: Total Marks = 347

Student 3: Total Marks = 257

Student 2: Total Marks = 254

Student 4: Total Marks = 158

`2. Reverse Words in a Sentence (Using Strings)`

`Problem Statement:`

Write a function to `reverse the words` in a sentence while keeping word order intact.

Example:

- Input: `"Hello World"`

- Output: `"olleH dlroW"`

```c
#include <stdio.h>

#include <string.h>



void reverseWord(char *start, char *end) {

    while (start < end) {

        char temp = *start;

        *start = *end;

        *end = temp;

        start++;

        end--;

    }

}



void reverseWords(char str[]) {
```

```
    int len = strlen(str);
    int start = 0;

    for (int i = 0; i <= len; i++) {
        if (str[i] == ' ' || str[i] == '\0') {
            reverseWord(&str[start], &str[i - 1]);
            start = i + 1;
        }
    }
}

int main() {
    char str[1000]; // Assuming a maximum sentence length of 1000 characters

    printf("Enter a sentence: ");
    fgets(str, sizeof(str), stdin);  // Use fgets to handle spaces correctly
    str[strcspn(str, "\n")] = 0;

    reverseWords(str);

    printf("Reversed Sentence: %s\n", str);
```

```
    return 0;

}
```

Enter a sentence: my name is ryan

Reversed Sentence: ym eman si nayr

`3. Check If a String Is a Palindrome`

`Problem Statement:`

A `palindrome` is a word or sentence that reads the same forward and backward.

Write a program to check if a given string is a `palindrome (ignoring case & spaces)`.

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>



char *strrev(char *str) {

    int len = strlen(str);



      // Temporary char array to store the

    // reversed string

    char *rev = (char *)malloc

      (sizeof(char) * (len + 1));



    // Reversing the string

    for (int i = 0; i < len; i++) {

        rev[i] = str[len - i - 1];
```

```
    }

    rev[len] = '\0';

    return rev;

}


void isPalindrome(char *str) {


    // Reversing the string

    char *rev = strrev(str);


    // Check if the original and reversed

      // strings are equal

    if (strcmp(str, rev) == 0)

        printf("\"%s\" is palindrome.\n",

               str);

    else

        printf("\"%s\" is not palindrome.\n",

               str);

}


int main() {


      // Cheking for palindrome strings
```

```
    isPalindrome("Ryan");

      isPalindrome("11111");



    return 0;

}
```

"Ryan" is not palindrome.

"11111" is palindrome.

`4. Find Most Frequent Element in an Array`

`Problem Statement:`

Given an array of `N integers`, find the `most frequently occurring` number.

- If multiple numbers occur with the same frequency, return the `smallest` one.

```
#include <stdio.h>

#include <limits.h> // For INT_MAX



int mostFrequentElement(int arr[], int n) {

    int maxFreq = 0;

    int mostFreqElement = INT_MAX; // Initialize with the largest possible
integer



    for (int i = 0; i < n; i++) {

        int currentFreq = 0;

        for (int j = 0; j < n; j++) {
```

```
            if (arr[i] == arr[j]) {
                currentFreq++;
            }
        }

        if (currentFreq > maxFreq) {
            maxFreq = currentFreq;
            mostFreqElement = arr[i];
        } else if (currentFreq == maxFreq && arr[i] < mostFreqElement) {
            mostFreqElement = arr[i]; // Choose the smaller one if
frequencies are equal
        }
    }
    printf("Most Frequent Element: %d (occurs %d times)\n",
mostFreqElement,maxFreq);
    return mostFreqElement;
}

int main() {
    int n;

    printf("Enter array size: ");
    scanf("%d", &n);
```

```
    int arr[n]; // Using Variable Length Array (VLA) - C99 and later


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


    mostFrequentElement(arr, n);



    return 0;
}
```

Enter array size: 10

Enter elements: 1

2

3

4

6

7

8

9

0

1 Most Frequent Element: 1 (occurs 2 times)

`5. Find a Substring in a Given String`

`Problem Statement:`

Write a function to `find a substring` within a given string.

- Return the `position (index)` of the substring if found, otherwise return `-1`.

```c
#include <stdio.h>

#include <string.h>



int findSubstring(char str[], char sub[]) {

    int strLen = strlen(str);

    int subLen = strlen(sub);



    if (subLen > strLen) {

        return -1; // Substring cannot be longer than the main string

    }



    for (int i = 0; i <= strLen - subLen; i++) { // Iterate up to where substring can fit

        int j;

        for (j = 0; j < subLen; j++) {

            if (str[i + j] != sub[j]) {

                break; // Mismatch, break inner loop

            }

        }
```

```
        if (j == subLen) { // If inner loop completed without a break,
it's a match
            return i; // Return starting index of the substring
        }
    }

    return -1; // Substring not found
}

int main() {
    char str[1000];
    char sub[1000];

    printf("Enter main string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = 0; // Remove trailing newline

    printf("Enter substring: ");
    fgets(sub, sizeof(sub), stdin);
    sub[strcspn(sub, "\n")] = 0; // Remove trailing newline

    int index = findSubstring(str, sub);

    if (index != -1) {
```

```
        printf("Substring found at index: %d\n", index);
    } else {
        printf("Substring not found.\n");
    }

    return 0;
}
```

Enter main string: ryan

Enter substring: ry

Substring found at index: 0

ryan@20245