Find the maximum difference between any two elements in the array

```
#include <stdio.h>
int maxDiff(int arr[], int n) {
    if (n < 2) {
        return -1;
    }
    int maxDiff = arr[1] - arr[0];
    int minElement = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] - minElement > maxDiff) {
            maxDiff = arr[i] - minElement;
        }
        if (arr[i] < minElement) {
            minElement = arr[i];
        }
    }
    return maxDiff;
}
int main() {
    return 0;
}
```

OUTPUT:

```
The given array is :  7  9  5  6  13  2
The elements which provide maximum difference is: 5,   13
The Maximum difference between two elements in the array is:  8
```

Given an array of integers, find the longest increasing subarray.

```
#include <stdio.h>

void longestIncreasingSubarray(int arr[], int n) {
    int maxLength = 1;
    int currentLength = 1;
    int startIndex = 0;
    int maxStartIndex = 0;

    for (int i = 1; i < n; i++) {
        if (arr[i] > arr[i - 1]) {
            currentLength++;
            if (currentLength > maxLength) {
                maxLength = currentLength;
                maxStartIndex = startIndex;
            }
        } else {
            currentLength = 1;
            startIndex = i;
        }
    }

    printf("Longest increasing subarray: ");
    for (int i = maxStartIndex; i < maxStartIndex + maxLength; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {1, 3, 8, 4, 5, 6, 3, 4, 5, 9, 10, 11};
    int n = sizeof(arr) / sizeof(arr[0]);
    longestIncreasingSubarray(arr, n);
    return 0;
}
```

OUTPUT:

```
Longest increasing subarray: 3 4 5 9 10 11
```

Find the common elements between them

```
#include <stdio.h>

void findCommonElements(int arr1[], int size1, int arr2[], int size2) {
    printf("Common elements: ");
    for (int i = 0; i < size1; i++) {
        for (int j = 0; j < size2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d ", arr1[i]);
                break;
            }
        }
    }
    printf("\n");
}

int main() {
    int arr1[] = {1, 3, 5, 7, 9};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);

    int arr2[] = {2, 4, 6, 8, 10, 7};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    findCommonElements(arr1, size1, arr2, size2);
    return 0;
}
```

OUTPUT:

Common elements: 7

ANALYTICAL SESSION:

Longest substring without repeating characters in a given string.

```
#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 100
int longestSubstringWithoutRepeating(char *str) {
    int n = strlen(str);
    int visited[256] = {0};
    int maxLength = 0;
    int start = 0;
    for (int i = 0; i < n; i++) {
        start = (visited[str[i]] > start) ? visited[str[i]] : start;
        visited[str[i]] = i + 1;
        maxLength = (i - start + 1 > maxLength) ? i - start + 1 : maxLengt;
    }
    return maxLength;
}
int main() {
    char str[MAX_LENGTH];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    int length = longestSubstringWithoutRepeating(str);
    printf(" longest substring without repeating: %d\n", length);
    return 0;
}
```

OUTPUT:

```
Enter a string: Yerriswamy
The length of the longest substring without repeating characters is: 7
```

Find the maximum product subarray in a given array of integers:

```
#include <stdio.h>
int maxProductSubarray(int arr[], int n) {
    if (n == 0) {
        return 0;
    }
    int maxEndingHere = arr[0];
    int minEndingHere = arr[0];
    int maxSoFar = arr[0];
    for (int i = 1; i < n; i++) {
        int temp = maxEndingHere;
        maxEndingHere = maximum(arr[i], maximum(arr[i] * maxEndingHere, arr[i] *
minEndingHere));
        minEndingHere = minimum(arr[i], minimum(arr[i] * temp, arr[i] * minEndingHere));
        maxSoFar = maximum(maxSoFar, maxEndingHere);
    }
    return maxSoFar;
}
int maximum(int a, int b) {
    return (a > b) ? a : b;
}
int minimum(int a, int b) {
    return (a < b) ? a : b;
}
int main() {
    int arr[] = {2, 3, -2, 4};
    int n = sizeof(arr) / sizeof(arr[0]);
    int maxProduct = maxProductSubarray(arr, n);
    printf("Maximum product subarray: %d\n", maxProduct);
    return 0;
}
```

OUTPUT:

```
Maximum product subarray: 6
```