11. Write a C program that takes user input for an array of strings and returns the first palindromic string.

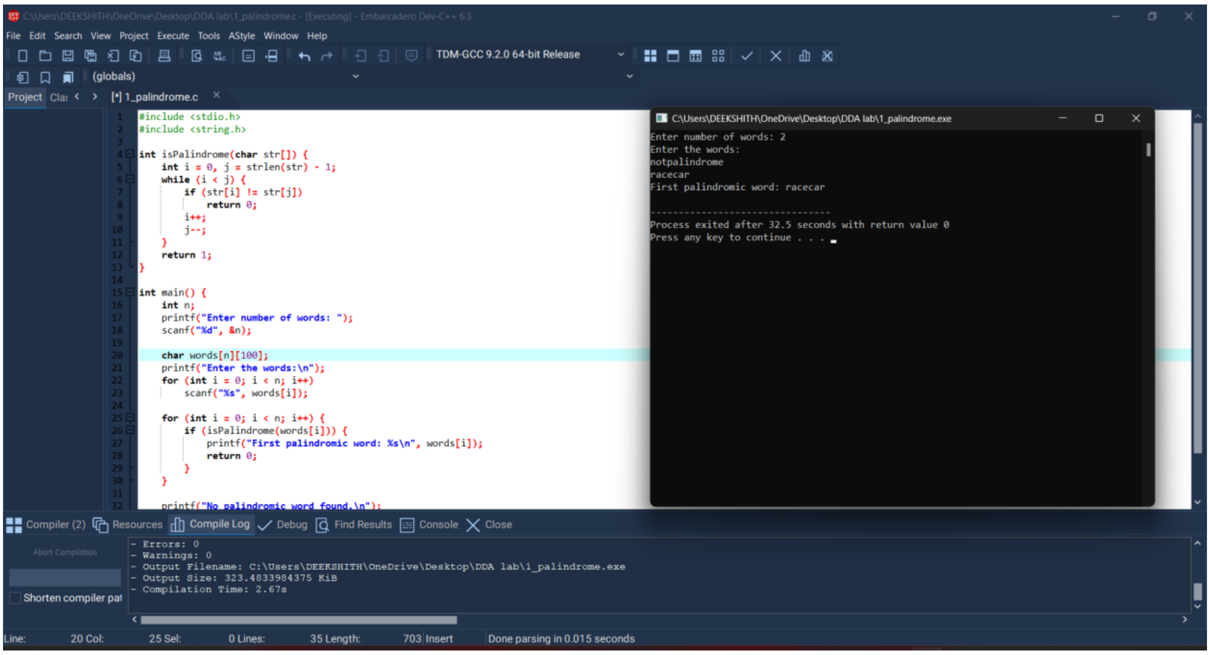
Aim: To find and display the first palindromic string in a given array of words.

Algorithm:

1. start
2. Input number of words n.
3. Read n words into an array.
4. For each word:
5. Check if it is a palindrome (compare characters from start and end).
6. If found, print it and stop.
7. If no palindrome is found, print an empty string ("").
8. End

Input: racecar

Output: is palindrome


C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\1_palindrome.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
[*] 1_palindrome.c
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 20 Col: 25 Sel: 0 Lines: 35 Length: 703 Insert Done parsing in 0.015 seconds

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

int isPalindrome(char str[]) {
    int i = 0, j = strlen(str) - 1;
    while (i < j) {
        if (str[i] != str[j])
            return 0;
        i++;
        j--;
    }
    return 1;
}

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

    char words[n][100];
    printf("Enter the words:\n");
    for (int i = 0; i < n; i++)
        scanf("%s", words[i]);

    for (int i = 0; i < n; i++) {
        if (isPalindrome(words[i])) {
            printf("First palindromic word: %s\n", words[i]);
            return 0;
        }
    }

    printf("No palindromic word found.\n");
```

C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\1_palindrome.exe

```
Enter number of words: 2
Enter the words:
notpalindrome
racecar
First palindromic word: racecar

--------------------------------
Process exited after 32.5 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\1_palindrome.exe
- Output Size: 323.4833984375 KiB
- Compilation Time: 2.67s
```

12. Program: Common elements count between two arrays

Aim:

Find how many elements of nums1 exist in nums2, and vice versa.

Algorithm:

1. Input n and m, and arrays nums1, nums2.
2. For each element in nums1, check if it appears in nums2 → increment answer1.
3. Do the same in reverse for answer2.
4. Print [answer1, answer2]

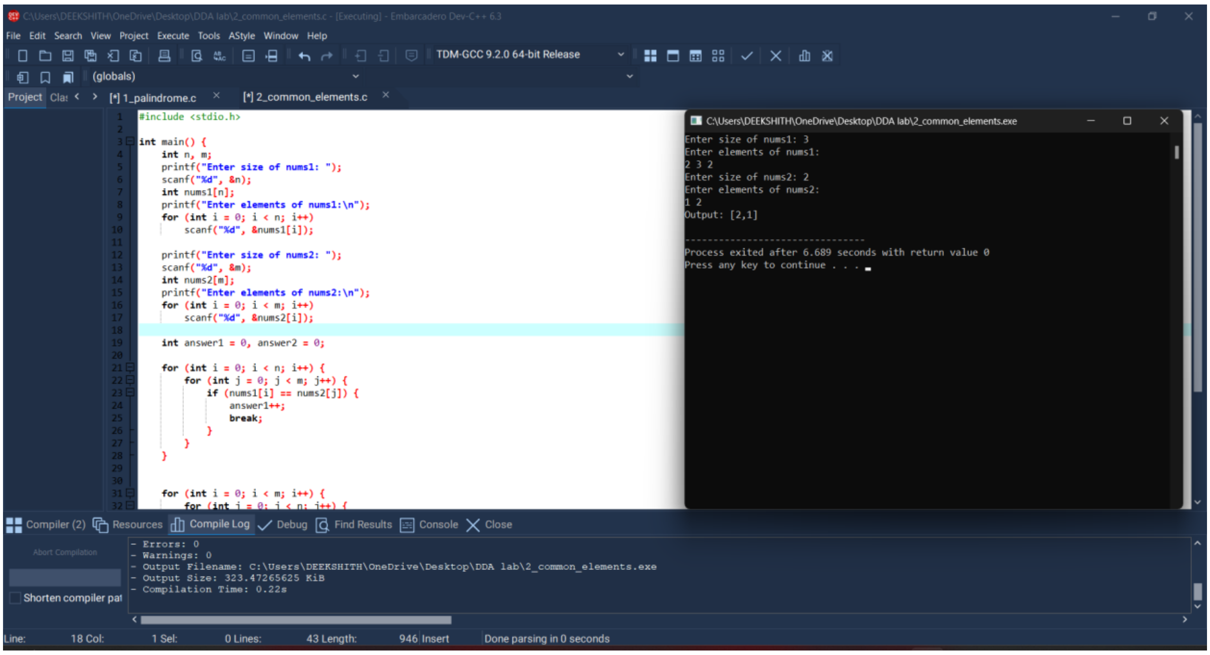
Input: Enter size of num1: 3

Enter elements of num1**:** 2 3 2

Enter size of num2: 2

Enter elements of num1**:** 1 2

Output: [2, 1]


C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\2_common_elements.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
[*] 1_palindrome.c
[*] 2_common_elements.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\2_common_elements.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 18 Col: 1 Sel: 0 Lines: 43 Length: 946 Insert
Done parsing in 0 seconds

```c
#include <stdio.h>

int main() {
    int n, m;
    printf("Enter size of nums1: ");
    scanf("%d", &n);
    int nums1[n];
    printf("Enter elements of nums1:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &nums1[i]);

    printf("Enter size of nums2: ");
    scanf("%d", &m);
    int nums2[m];
    printf("Enter elements of nums2:\n");
    for (int i = 0; i < m; i++)
        scanf("%d", &nums2[i]);

    int answer1 = 0, answer2 = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (nums1[i] == nums2[j]) {
                answer1++;
                break;
            }
        }
    }


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

```
Enter size of nums1: 3
Enter elements of nums1:
2 3 2
Enter size of nums2: 2
Enter elements of nums2:
1 2
Output: [2,1]

--------------------------------
Process exited after 6.689 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\2_common_elements.exe
- Output Size: 323.47265625 KiB
- Compilation Time: 0.22s
```

13. Program: Sum of squares of distinct counts of all subarrays

Aim:

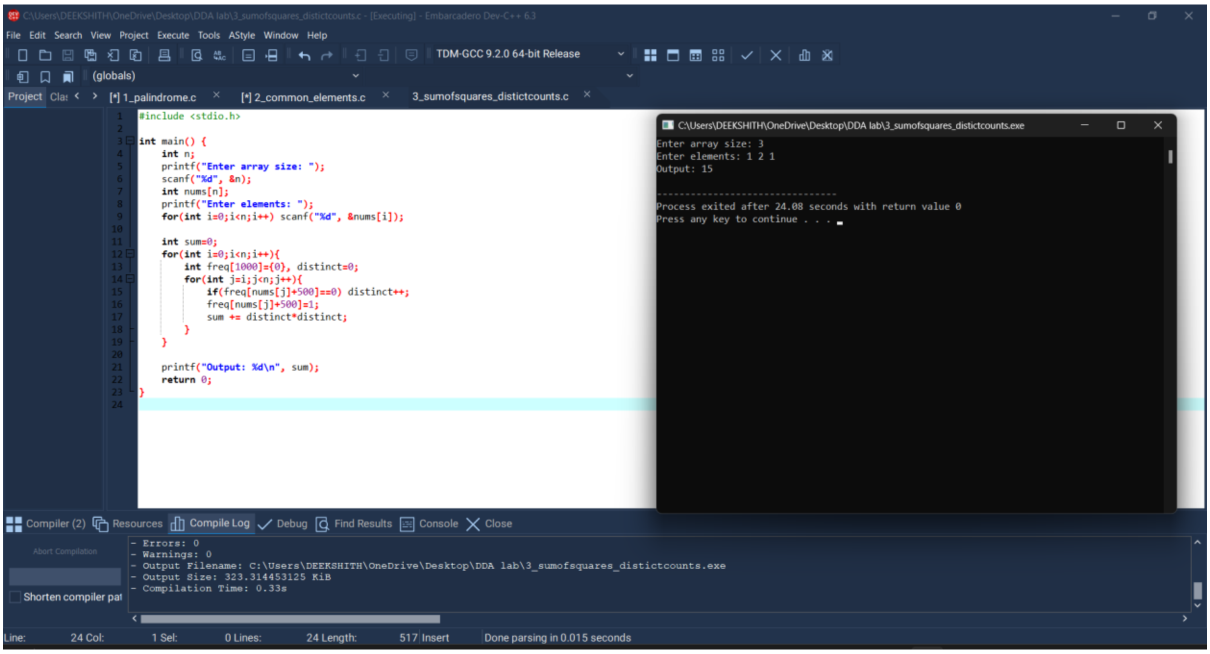
Find sum of squares of distinct element counts in all subarrays.

Algorithm:

1. Generate all subarrays.
2. Count distinct elements in each subarray.
3. Add (distinct_count²) to sum.

Input: [1,2,1]

Output: 15



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\3_sumofsquares_distictcounts.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
[*] 1_palindrome.c
[*] 2_common_elements.c
3_sumofsquares_distictcounts.c

```c
#include <stdio.h>

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

    int sum=0;
    for(int i=0;i<n;i++){
        int freq[1000]={0}, distinct=0;
        for(int j=i;j<n;j++){
            if(freq[nums[j]+500]==0) distinct++;
            freq[nums[j]+500]=1;
            sum += distinct*distinct;
        }
    }

    printf("Output: %d\n", sum);
    return 0;
}
```

```
Enter array size: 3
Enter elements: 1 2 1
Output: 15

--------------------------------
Process exited after 24.08 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\3_sumofsquares_distictcounts.exe
- Output Size: 323.314453125 KiB
- Compilation Time: 0.33s
```

Line: 24 Col: 1 Sel: 0 Lines: 24 Length: 517 Insert Done parsing in 0.015 seconds

14._Program: Count pairs satisfying given conditions

Aim:

Find number of pairs (i,j) such that nums[i] == nums[j] and (i*j) divisible by k.

Algorithm:

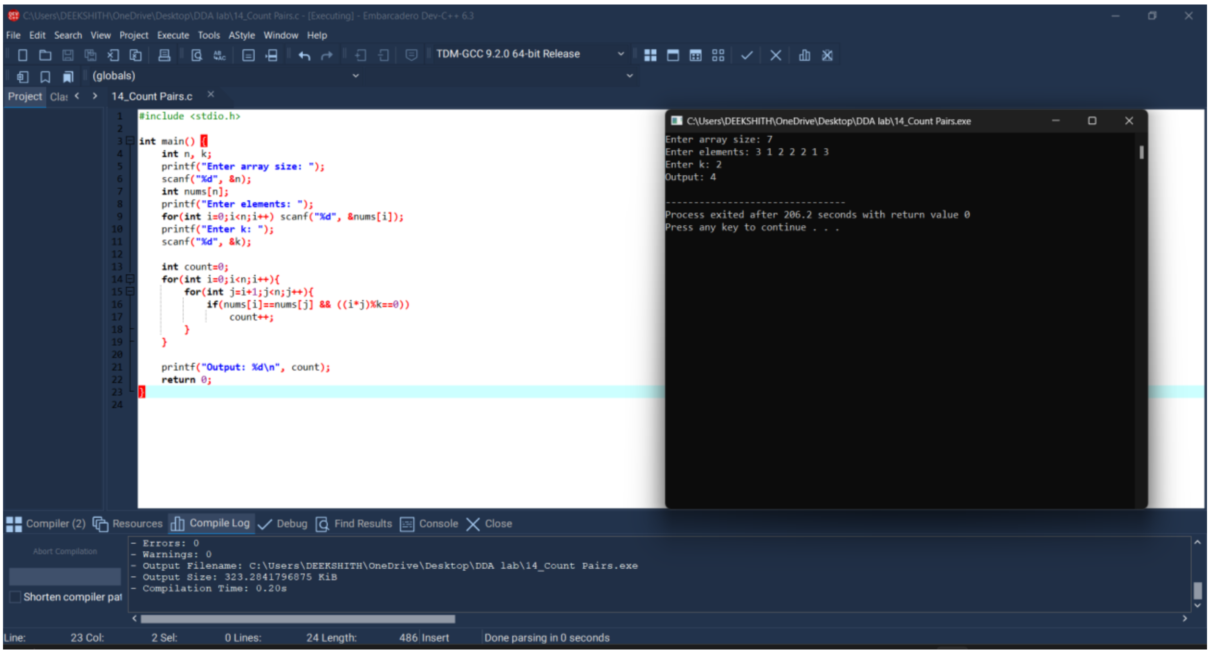
1. Take input n, k.

2. Check all pairs (i,j) with i < j.

3. If conditions satisfy, increment count.

Input: nums = [3,1,2,2,2,1,3], **k=2**

Output: 4



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\14_Count Pairs.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\14_Count Pairs.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 23 Col: 2 Sel: 0 Lines: 24 Length: 486 Insert Done parsing in 0 seconds

```c
#include <stdio.h>

int main() {
    int n, k;
    printf("Enter array size: ");
    scanf("%d", &n);
    int nums[n];
    printf("Enter elements: ");
    for(int i=0;i<n;i++) scanf("%d", &nums[i]);
    printf("Enter k: ");
    scanf("%d", &k);

    int count=0;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            if(nums[i]==nums[j] && ((i*j)%k==0))
                count++;
        }
    }

    printf("Output: %d\n", count);
    return 0;
}
```

```
Enter array size: 7
Enter elements: 3 1 2 2 2 1 3
Enter k: 2
Output: 4

--------------------------------
Process exited after 206.2 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\14_Count Pairs.exe
- Output Size: 323.2841796875 KiB
- Compilation Time: 0.20s
```

15._Program: Find Maximum Element

Aim:

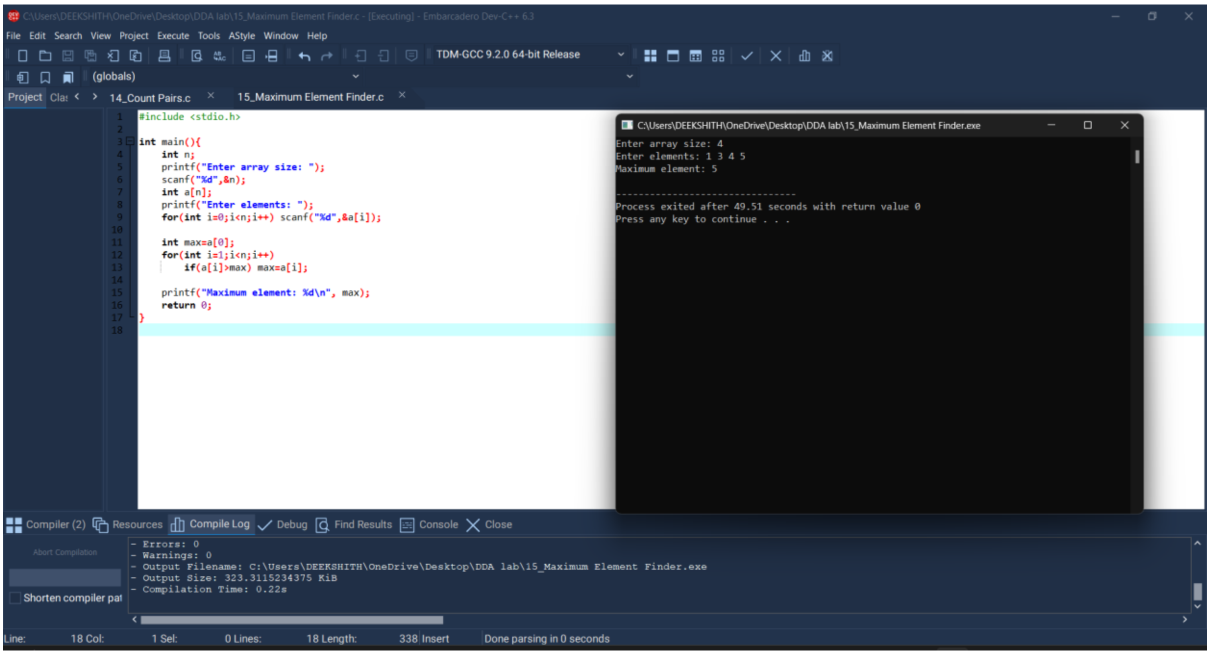
Find the maximum element in an array.

Algorithm:

1. Input array.
2. Traverse and track maximum.
3. Print max.

Input: 1 3 4 5

Output: 5



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\15_Maximum Element Finder.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c

```c
#include <stdio.h>

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

    int max=a[0];
    for(int i=1;i<n;i++)
        if(a[i]>max) max=a[i];

    printf("Maximum element: %d\n", max);
    return 0;
}
```

```
Enter array size: 4
Enter elements: 1 3 4 5
Maximum element: 5

--------------------------------
Process exited after 49.51 seconds with return value 0
Press any key to continue . . .
```

Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\15_Maximum Element Finder.exe
- Output Size: 323.3115234375 KiB
- Compilation Time: 0.22s
```

Line: 18 Col: 1 Sel: 0 Lines: 18 Length: 338 Insert Done parsing in 0 seconds

16. Program: Sort array and find maximum

Aim:

Sort list efficiently and return the max element.

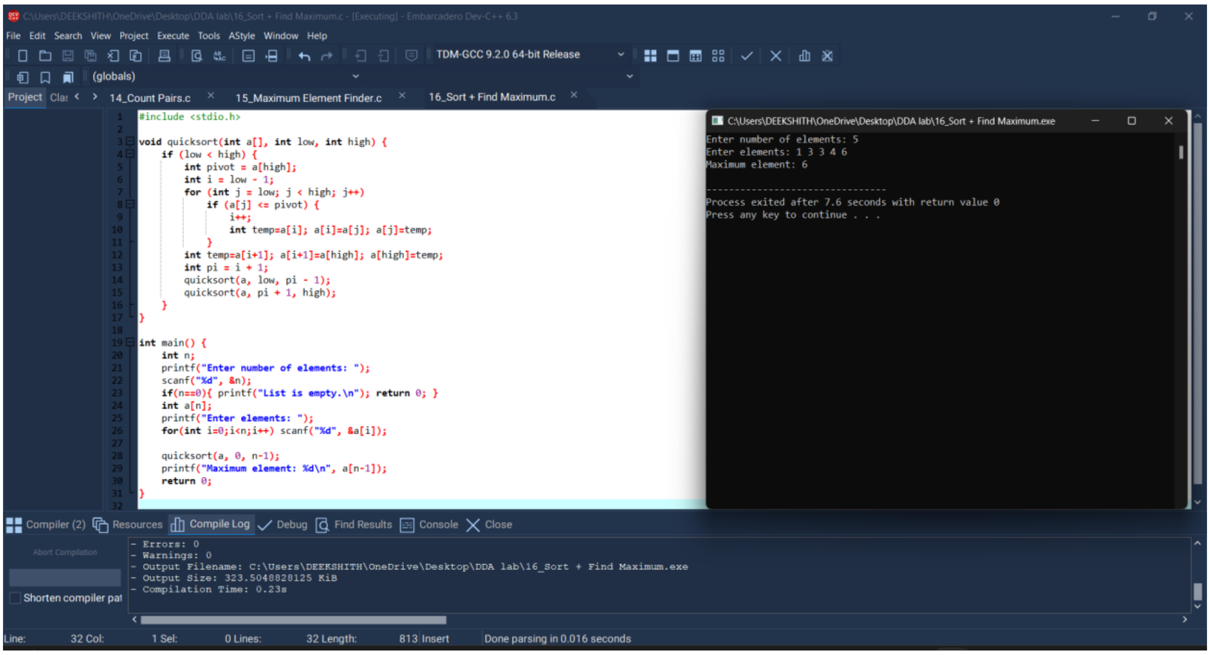
Algorithm:

1. Input array.
2. Use efficient sort (e.g., quicksort).
3. Return last element as max.

Input: 1 4 3 5 2

Output: Sorted order: 1 2 3 4 5

maximum element: 5



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\16_Sort + Find Maximum.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c
16_Sort + Find Maximum.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\16_Sort + Find Maximum.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 32 Col: 1 Sel: 0 Lines: 32 Length: 813 Insert Done parsing in 0.016 seconds

```c
#include <stdio.h>

void quicksort(int a[], int low, int high) {
    if (low < high) {
        int pivot = a[high];
        int i = low - 1;
        for (int j = low; j < high; j++)
            if (a[j] <= pivot) {
                i++;
                int temp=a[i]; a[i]=a[j]; a[j]=temp;
            }
        int temp=a[i+1]; a[i+1]=a[high]; a[high]=temp;
        int pi = i + 1;
        quicksort(a, low, pi - 1);
        quicksort(a, pi + 1, high);
    }
}

int main() {
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    if(n==0){ printf("List is empty.\n"); return 0; }
    int a[n];
    printf("Enter elements: ");
    for(int i=0;i<n;i++) scanf("%d", &a[i]);

    quicksort(a, 0, n-1);
    printf("Maximum element: %d\n", a[n-1]);
    return 0;
}
```

```
Enter number of elements: 5
Enter elements: 1 3 3 4 6
Maximum element: 6

--------------------------------
Process exited after 7.6 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\16_Sort + Find Maximum.exe
- Output Size: 323.5048828125 KiB
- Compilation Time: 0.23s
```

17. Program: Unique elements

Aim:

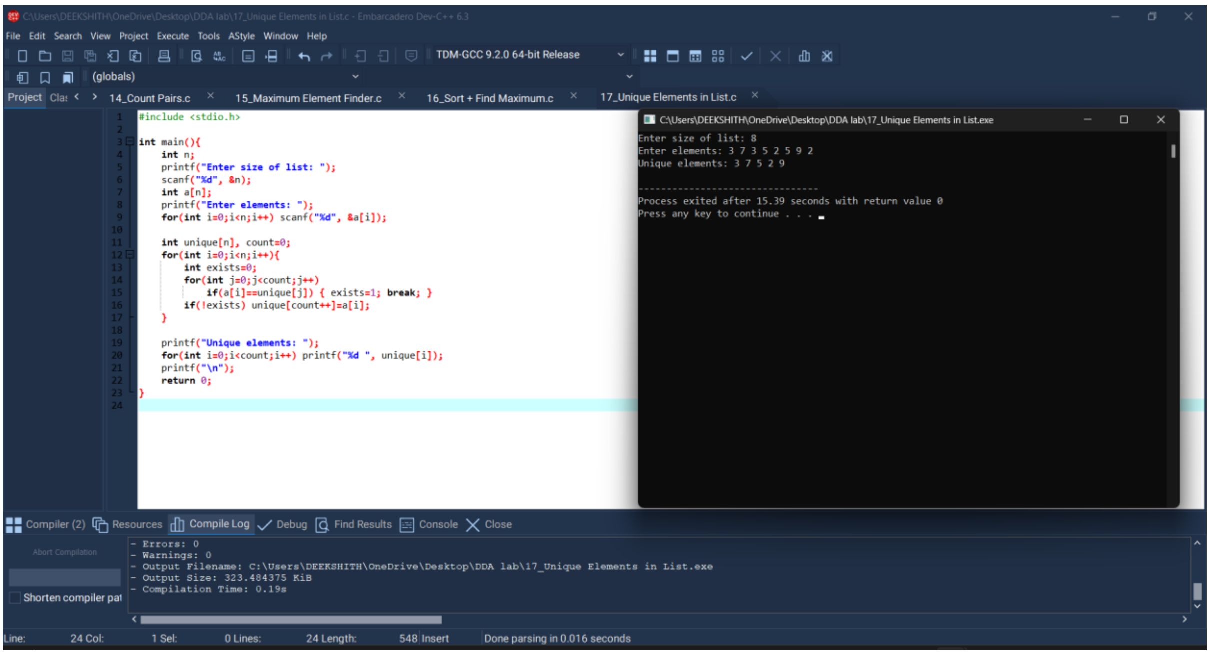
Create new list with unique elements only.

Algorithm:

1. Input array.
2. For each element, check if already exists in new array.
3. Print unique list.

Input: enter elements: 3 7 3 5 2 5 9 2

Output: 3 7 5 2 9



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\17_Unique Elements in List.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c
16_Sort + Find Maximum.c
17_Unique Elements in List.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\17_Unique Elements in List.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 24 Col: 1 Sel: 0 Lines: 24 Length: 548 Insert Done parsing in 0.016 seconds

```c
#include <stdio.h>

int main(){
    int n;
    printf("Enter size of list: ");
    scanf("%d", &n);
    int a[n];
    printf("Enter elements: ");
    for(int i=0;i<n;i++) scanf("%d", &a[i]);

    int unique[n], count=0;
    for(int i=0;i<n;i++){
        int exists=0;
        for(int j=0;j<count;j++)
            if(a[i]==unique[j]) { exists=1; break; }
        if(!exists) unique[count++]=a[i];
    }

    printf("Unique elements: ");
    for(int i=0;i<count;i++) printf("%d ", unique[i]);
    printf("\n");
    return 0;
}
```

```
Enter size of list: 8
Enter elements: 3 7 3 5 2 5 9 2
Unique elements: 3 7 5 2 9

--------------------------------
Process exited after 15.39 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\17_Unique Elements in List.exe
- Output Size: 323.484375 KiB
- Compilation Time: 0.19s
```

18. Program: Bubble Sort

Aim:
Sort an array using bubble sort.

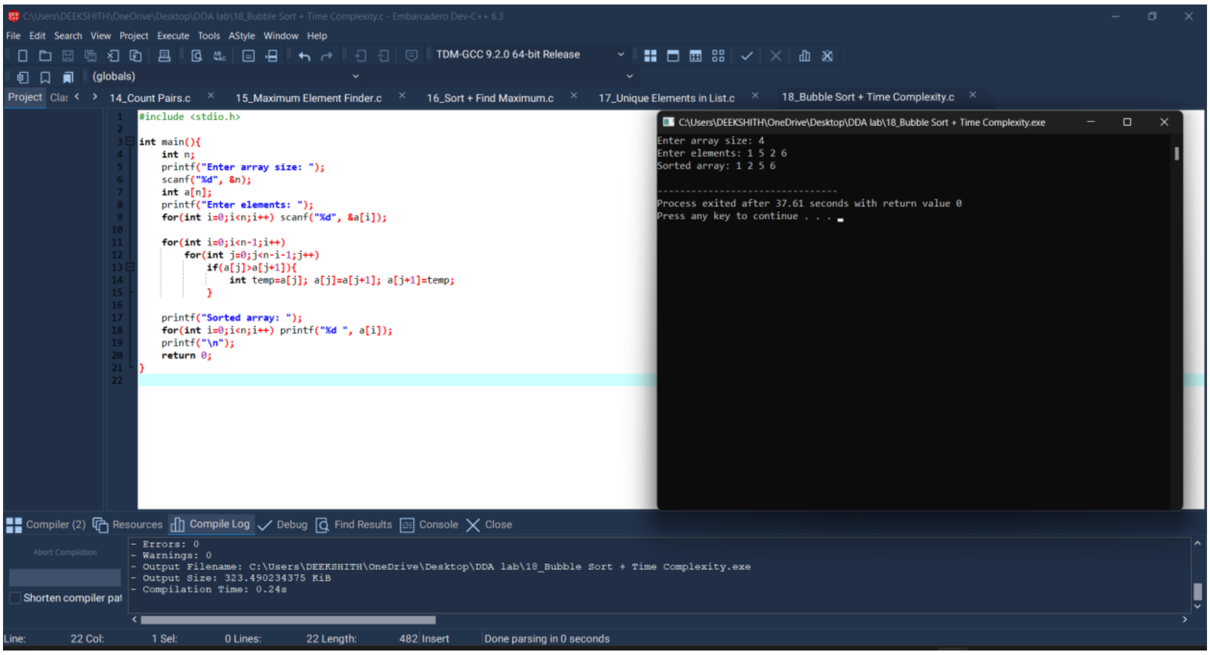
Algorithm:

1. Start
2. Repeatedly swap adjacent elements if out of order.
3. Stop

Input: Enter array size: 4

Enter elements: 1 5 2 6

Output: sorted array: 1 2 5 6



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\18_Bubble Sort + Time Complexity.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c
16_Sort + Find Maximum.c
17_Unique Elements in List.c
18_Bubble Sort + Time Complexity.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\18_Bubble Sort + Time Complexity.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 22 Col: 1 Sel: 0 Lines: 22 Length: 482 Insert Done parsing in 0 seconds

```c
#include <stdio.h>

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

    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(a[j]>a[j+1]){
                int temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;
            }

    printf("Sorted array: ");
    for(int i=0;i<n;i++) printf("%d ", a[i]);
    printf("\n");
    return 0;
}
```

```
Enter array size: 4
Enter elements: 1 5 2 6
Sorted array: 1 2 5 6

--------------------------------
Process exited after 37.61 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\18_Bubble Sort + Time Complexity.exe
- Output Size: 323.490234375 KiB
- Compilation Time: 0.24s
```

19. Program: Binary Search

Aim:

Search element in sorted array.

Algorithm:

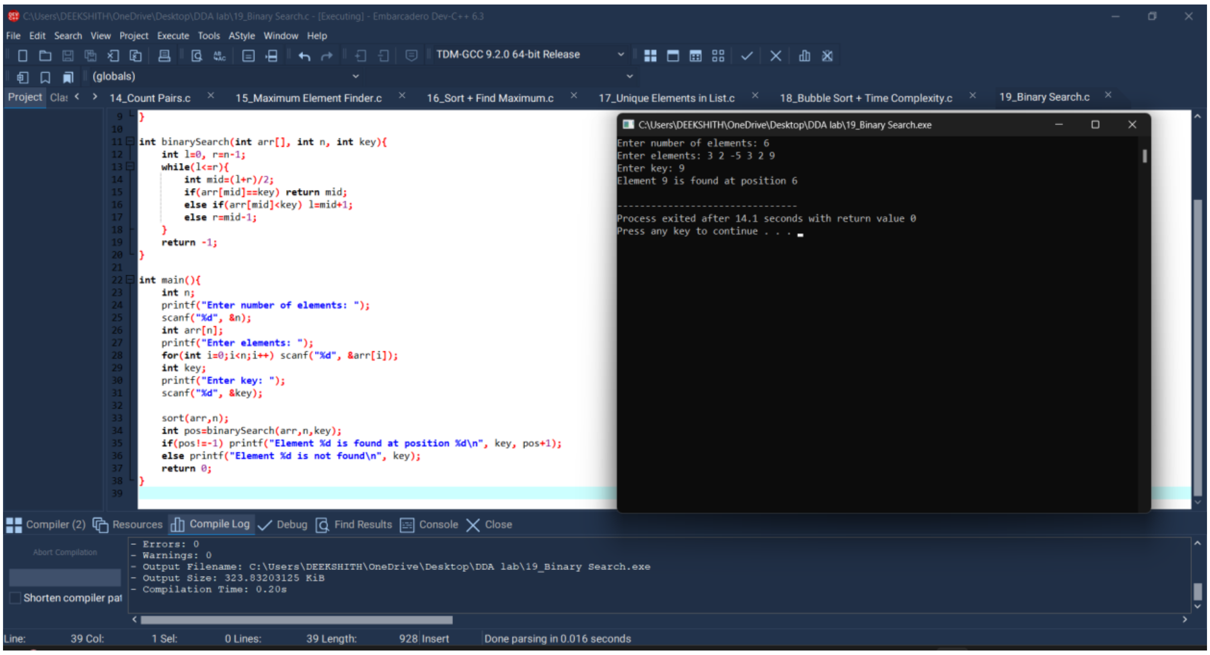
1. Start
2. Sort array.
3. Use binary search.
4. Return position or not found.
5. Stop

Input: Enter number of elements: 6

Enter elements: 3 2 -5 3 2 9

Enter key: 9

Output: Element 9 is found at position 6


C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\19_Binary Search.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c
16_Sort + Find Maximum.c
17_Unique Elements in List.c
18_Bubble Sort + Time Complexity.c
19_Binary Search.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\19_Binary Search.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 39 Col: 1 Sel: 0 Lines: 39 Length: 928 Insert Done parsing in 0.016 seconds

```c
}

int binarySearch(int arr[], int n, int key){
    int l=0, r=n-1;
    while(l<=r){
        int mid=(l+r)/2;
        if(arr[mid]==key) return mid;
        else if(arr[mid]<key) l=mid+1;
        else r=mid-1;
    }
    return -1;
}

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

    sort(arr,n);
    int pos=binarySearch(arr,n,key);
    if(pos!=-1) printf("Element %d is found at position %d\n", key, pos+1);
    else printf("Element %d is not found\n", key);
    return 0;
}
```

```
Enter number of elements: 6
Enter elements: 3 2 -5 3 2 9
Enter key: 9
Element 9 is found at position 6

--------------------------------
Process exited after 14.1 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\19_Binary Search.exe
- Output Size: 323.83203125 KiB
- Compilation Time: 0.20s
```

20. Program: Merge Sort (O(n log n))

Aim:
Sort array in ascending order using merge sort.

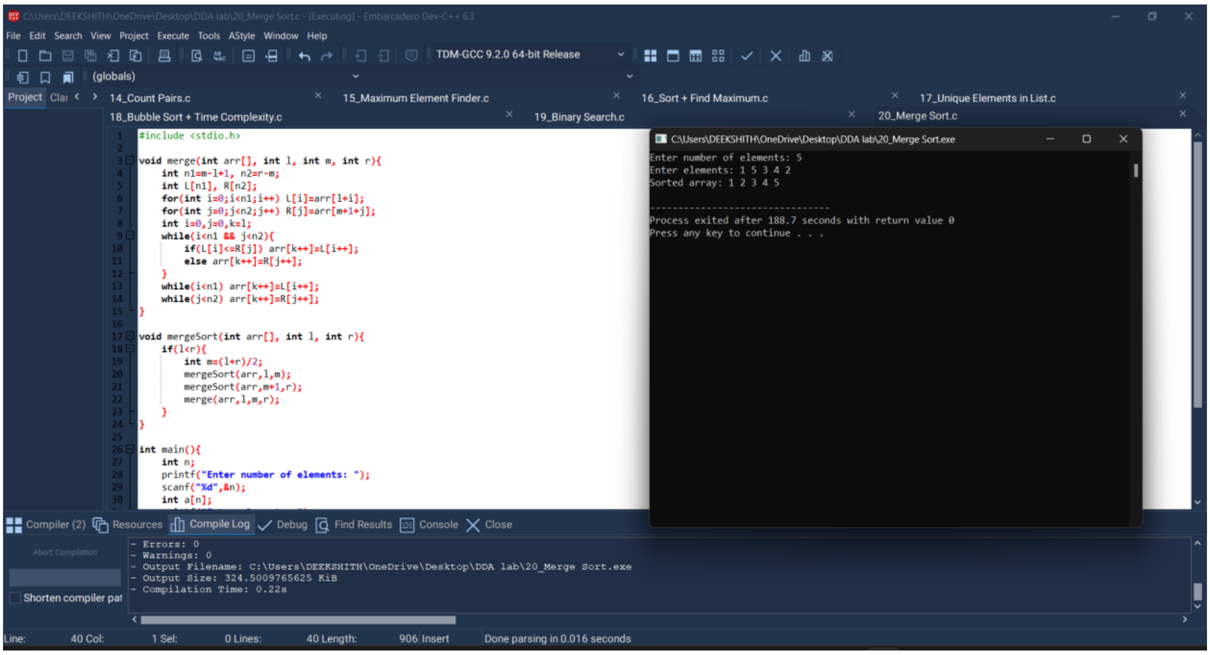
Algorithm:

1. Start
2. Divide array → Sort halves → Merge them.
3. Stop

Input: Enter array size: 5

Enter elements: 1 5 3 4 2

Output: sorted array: 1 2 3 4 5



C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\20_Merge Sort.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
14_Count Pairs.c
15_Maximum Element Finder.c
16_Sort + Find Maximum.c
17_Unique Elements in List.c
18_Bubble Sort + Time Complexity.c
19_Binary Search.c
20_Merge Sort.c
C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\20_Merge Sort.exe
Compiler (2) Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Line: 40 Col: 1 Sel: 0 Lines: 40 Length: 906 Insert Done parsing in 0.016 seconds

```c
#include <stdio.h>

void merge(int arr[], int l, int m, int r){
    int n1=m-l+1, n2=r-m;
    int L[n1], R[n2];
    for(int i=0;i<n1;i++) L[i]=arr[l+i];
    for(int j=0;j<n2;j++) R[j]=arr[m+1+j];
    int i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(L[i]<=R[j]) arr[k++]=L[i++];
        else arr[k++]=R[j++];
    }
    while(i<n1) arr[k++]=L[i++];
    while(j<n2) arr[k++]=R[j++];
}

void mergeSort(int arr[], int l, int r){
    if(l<r){
        int m=(l+r)/2;
        mergeSort(arr,l,m);
        mergeSort(arr,m+1,r);
        merge(arr,l,m,r);
    }
}

int main(){
    int n;
    printf("Enter number of elements: ");
    scanf("%d",&n);
    int a[n];
```

```
Enter number of elements: 5
Enter elements: 1 5 3 4 2
Sorted array: 1 2 3 4 5

--------------------------------
Process exited after 188.7 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\20_Merge Sort.exe
- Output Size: 324.5009765625 KiB
- Compilation Time: 0.22s
```