

**Program 1:**

```
#include <stdio.h>

#include <string.h>

int isPalindrome(char s[]) {

    int i = 0, j = strlen(s) - 1;

    while(i < j)

        if(s[i++] != s[j--]) return 0;

    return 1;

}

int main() {

    int n;

    printf("Enter number of words: ");

    scanf("%d", &n);

    char words[n][50];

    printf("Enter 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 string: %s\n", words[i]);

            return 0;

        }

    }

}
```

```
printf("No palindromic string found.\n");  
  
return 0;  
  
}
```

**Output:**

```
Enter number of words: 5  
Enter words:  
abc car ada racecar cool  
First palindromic string: ada
```

## Program 2:

```
#include <stdio.h>

int main() {

    int n, m, i, j, ans1 = 0, ans2 = 0;

    printf("Enter size of nums1 and nums2: ");

    scanf("%d %d", &n, &m);

    int nums1[n], nums2[m];

    printf("Enter elements of nums1: ");

    for(i = 0; i < n; i++) scanf("%d", &nums1[i]);

    printf("Enter elements of nums2: ");

    for(i = 0; i < m; i++) scanf("%d", &nums2[i]);

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

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

            if(nums1[i] == nums2[j]) { ans1++; break; }

    for(i = 0; i < m; i++)

        for(j = 0; j < n; j++)

            if(nums2[i] == nums1[j]) { ans2++; break; }

    printf("[%d, %d]\n", ans1, ans2);

    return 0;

}
```

### Output:

```
Enter size of nums1 and nums2: 5 6
Enter elements of nums1: 1 4 3 4 5
Enter elements of nums2: 4 5 7 6 3 2
[4, 3]
```

### Program 3:

```
#include <stdio.h>

int countDistinct(int arr[], int start, int end) {

    int count = 0, freq[100] = {0};

    for (int i = start; i <= end; i++)

        if (freq[arr[i]]++ == 0)

            count++;

    return count;
}

int main() {

    int n;

    printf("Enter size of array: ");

    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++)

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

        int d = countDistinct(nums, i, j);

        sum += d * d;

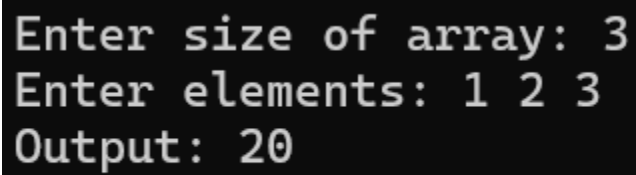
    }

printf("Output: %d\n", sum);

return 0;

}
```

#### **Output:**

A screenshot of a terminal window with a black background and light green text. It shows the input and output of the program. The first line is "Enter size of array: 3", the second line is "Enter elements: 1 2 3", and the third line is "Output: 20".

```
Enter size of array: 3
Enter elements: 1 2 3
Output: 20
```

#### **Program 4:**

```
#include <stdio.h>

int main() {

    int n, k, count = 0;

    printf("Enter size of array: ");

    scanf("%d", &n);
```

```
int nums[n];

printf("Enter elements: ");

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

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

printf("Enter value of k: ");

scanf("%d", &k);

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;

}
```

**Output:**

```
Enter size of array: 7
Enter elements: 3 2 2 2 2 1 3
Enter value of k: 2
Output: 6
```

### Program 5:

```
#include <stdio.h>

int main() {

    int n;

    printf("Enter size of array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter elements: ");

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

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

    int max = arr[0];

    for(int i = 1; i < n; i++)

        if(arr[i] > max)

            max = arr[i];

    printf("Maximum element: %d\n", max);

    return 0;

}
```

Output:

```
Enter size of array: 5
Enter elements: 1 2 3 4 5
Maximum element: 5
```

### Program 6:

```
#include <stdio.h>

void swap(int *a, int *b) {
    int t = *a; *a = *b; *b = t;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high], i = low - 1;
    for (int j = low; j < high; j++)
        if (arr[j] <= pivot) swap(&arr[++i], &arr[j]);
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

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



```
if (n == 0) {  
    printf("List is empty.\n");  
    return 0;  
}  
  
int arr[n];  
  
printf("Enter elements: ");  
  
for (int i = 0; i < n; i++) scanf("%d", &arr[i]);  
  
quickSort(arr, 0, n - 1);  
  
printf("Sorted list: ");  
  
for (int i = 0; i < n; i++) printf("%d ", arr[i]);  
  
printf("\nMaximum element: %d\n", arr[n - 1]);  
  
return 0;  
}
```

**Output:**

```
Enter size of list: 5  
Enter elements: 3 3 3 3 3  
Sorted list: 3 3 3 3 3  
Maximum element: 3
```

**Program 7:**

```
#include <stdio.h>

int main() {

    int n;

    printf("Enter size of list: ");

    scanf("%d", &n);

    int arr[n], unique[n], u = 0;

    printf("Enter elements: ");

    for (int i = 0; i < n; i++) scanf("%d", &arr[i]);

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

        int found = 0;

        for (int j = 0; j < u; j++)

            if (arr[i] == unique[j]) { found = 1; break; }

        if (!found) unique[u++] = arr[i];

    }

    printf("Unique elements: ");

    for (int i = 0; i < u; i++) printf("%d ", unique[i]);

    printf("\n");

    return 0;

}
```

**Output:**

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

**Program 8:**

```
#include <stdio.h>

int main() {
    int n, i, j, temp;

    printf("Enter number of elements: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter elements: ");

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

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

    for (i = 0; i < n - 1; i++)

        for (j = 0; j < n - i - 1; j++)

            if (arr[j] > arr[j + 1]) {

                temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;
            }
    }
```

```

    }

    printf("Sorted array: ");

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

        printf("%d ", arr[i]);

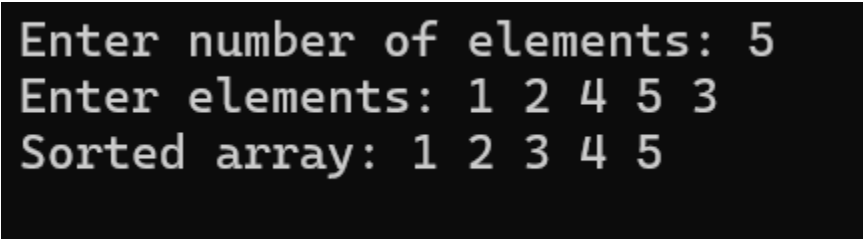
    printf("\n");

    return 0;

}

```

### Output:



```

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

```

### Program 9:

```

#include <stdio.h>

int main() {

    int n, i, key, low, high, mid, found = 0;

    printf("Enter number of elements: ");

    scanf("%d", &n);

    int arr[n];

```

```
printf("Enter %d sorted elements: ", n);

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

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

printf("Enter element to search: ");

scanf("%d", &key);

low = 0;

high = n - 1;

while (low <= high) {

    mid = (low + high) / 2;

    if (arr[mid] == key) {

        printf("Element %d is found at position %d\n", key, mid + 1);

        found = 1;

        break;

    } else if (arr[mid] < key)

        low = mid + 1;

    else

        high = mid - 1;

}

if (!found)

    printf("Element %d is not found\n", key);

return 0;

}
```

### Output:

```
Enter number of elements: 8
Enter 8 sorted elements: -9 3 4 6 8 9 10 30
Enter element to search: 10
Element 10 is found at position 7
```

### Program 10:

```
#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)
        arr[k++] = (L[i] <= R[j]) ? L[i++] : 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 arr[n];

    printf("Enter %d elements: ", n);

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

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

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: ");

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

        printf("%d ", arr[i]);

    printf("\n");

    return 0;

}

```

**Output:**

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