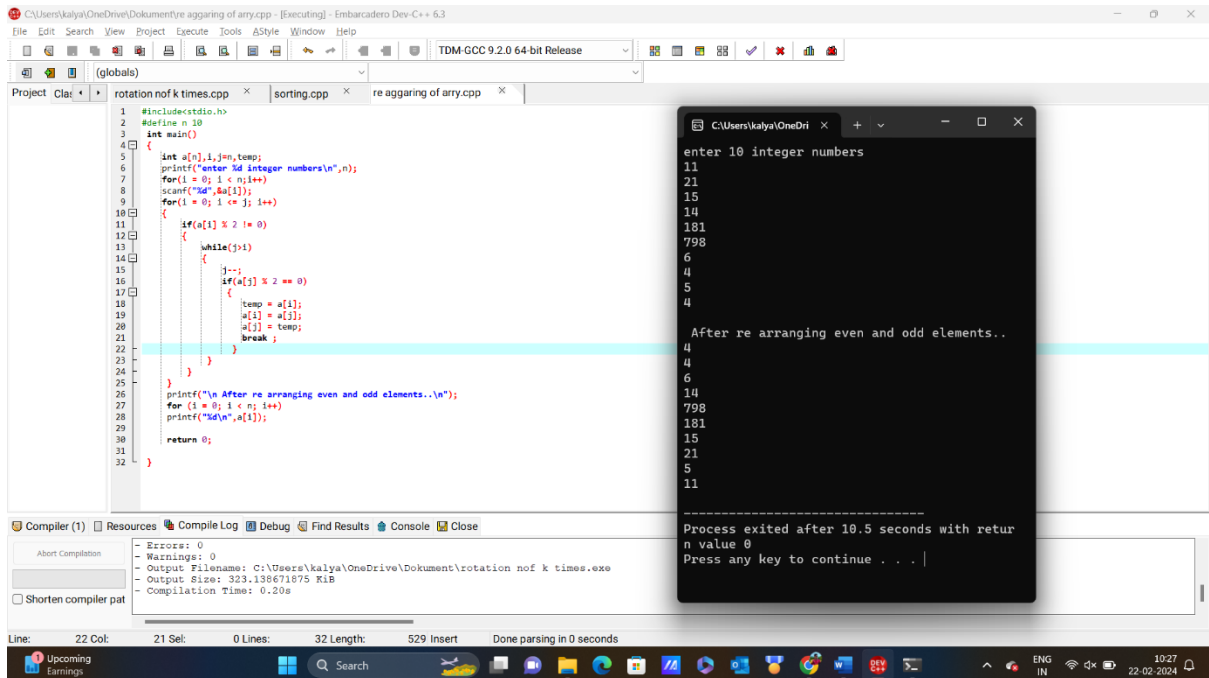


# C programming

## Day -2

### Rearranging of array by odd elements first then even elements

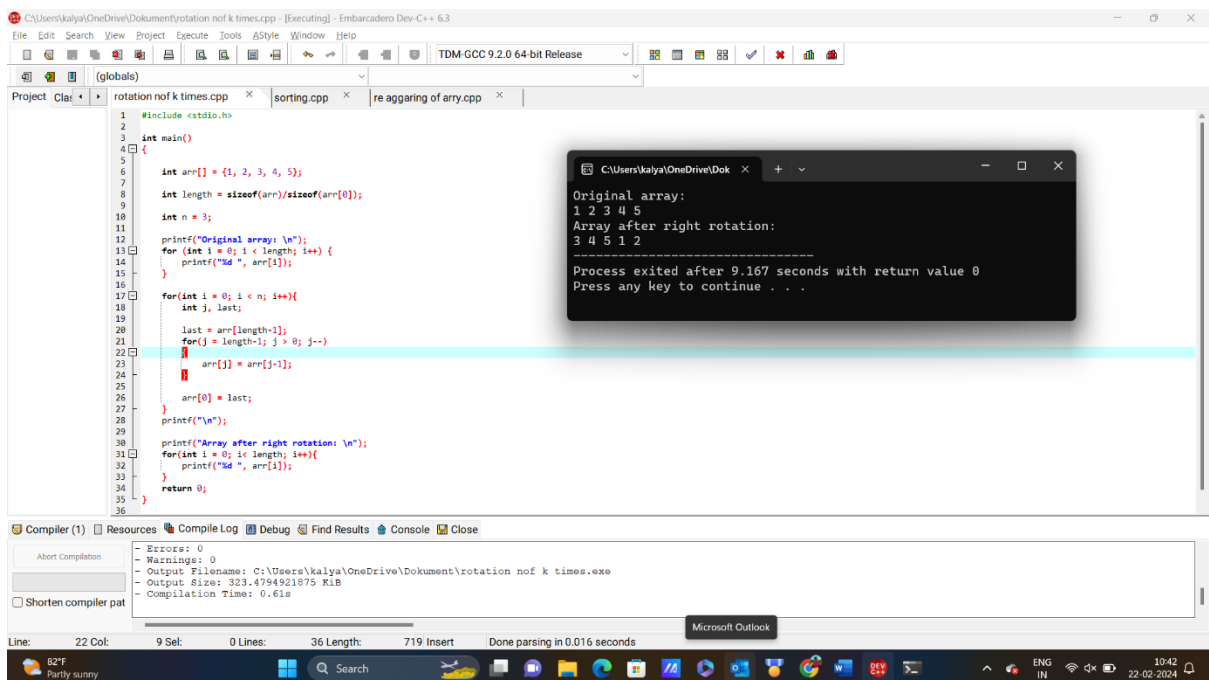


The screenshot shows an IDE with a C program that rearranges an array by placing odd elements first, followed by even elements. The code uses a two-pointer approach to swap elements. The console output shows the input array [11, 21, 15, 14, 181, 798, 6, 4, 5, 4] and the output array [4, 4, 6, 14, 798, 181, 15, 21, 5, 11] after rearranging even and odd elements.

```
#include <stdio.h>
#define n 10
int main()
{
    int a[n], i, j, temp;
    printf("enter %d integer numbers\n", n);
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n; i++)
    {
        if(a[i] % 2 != 0)
        {
            while(j < i)
            {
                j--;
                if(a[j] % 2 == 0)
                {
                    temp = a[j];
                    a[j] = a[i];
                    a[i] = temp;
                    break;
                }
            }
        }
    }
    printf("\n After re arranging even and odd elements..\n");
    for(i = 0; i < n; i++)
        printf("%d\n", a[i]);
    return 0;
}
```

enter 10 integer numbers  
11  
21  
15  
14  
181  
798  
6  
4  
5  
4  
After re arranging even and odd elements..  
4  
4  
6  
14  
798  
181  
15  
21  
5  
11  
Process exited after 10.5 seconds with return value 0  
Press any key to continue . . .

## 2. rotation of array in k times

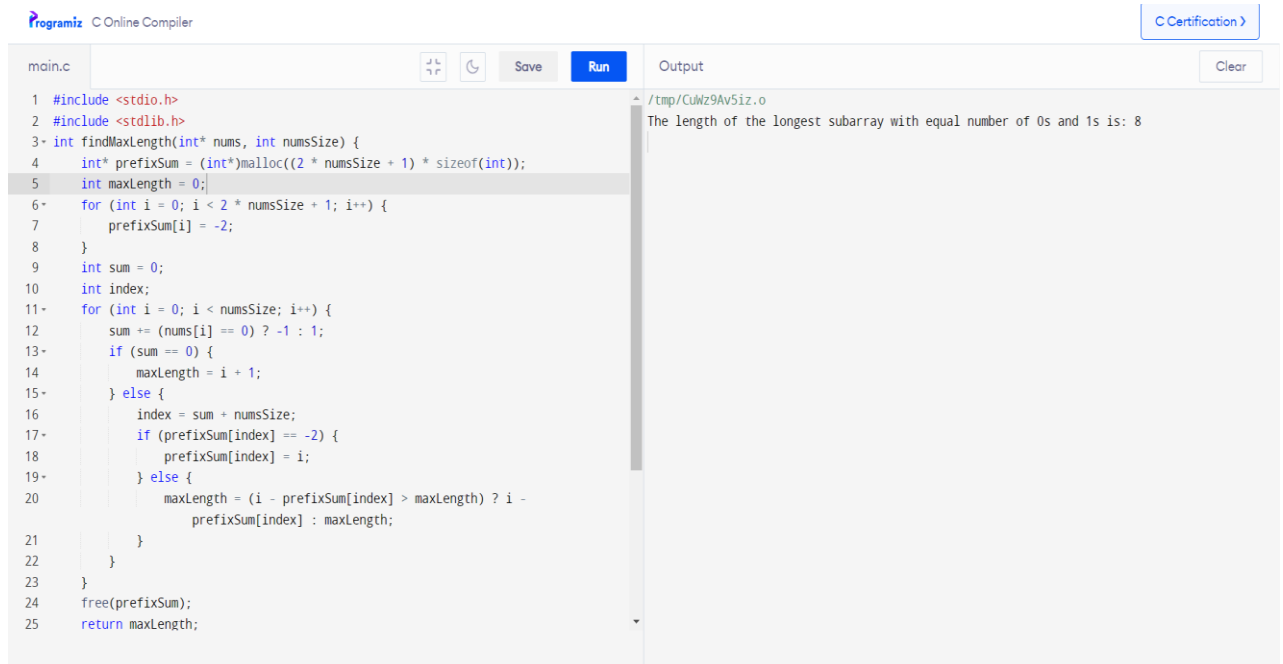


The screenshot shows an IDE with a C program that rotates an array in k times. The code uses a loop to rotate the array by one position at a time. The console output shows the original array [1, 2, 3, 4, 5] and the array after right rotation [3, 4, 5, 1, 2].

```
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int length = sizeof(arr)/sizeof(arr[0]);
    int n = 3;
    printf("Original array: \n");
    for(int i = 0; i < length; i++)
        printf("%d ", arr[i]);
    for(int i = 0; i < n; i++)
    {
        int j, last;
        last = arr[length-1];
        for(j = length-1; j > 0; j--)
            arr[j] = arr[j-1];
        arr[0] = last;
        printf("\n");
    }
    printf("Array after right rotation: \n");
    for(int i = 0; i < length; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Original array:  
1 2 3 4 5  
Array after right rotation:  
3 4 5 1 2  
Process exited after 9.167 seconds with return value 0  
Press any key to continue . . .

### 3.largest subarray in 0s and 1s

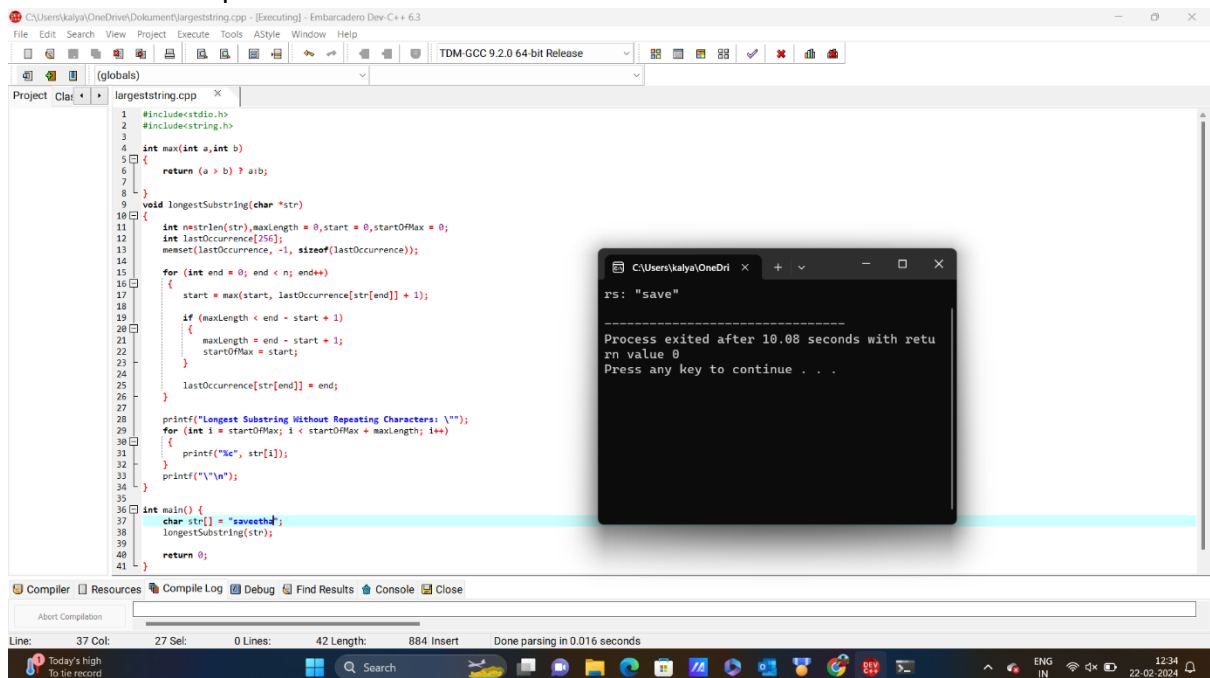


```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 int findMaxLength(int* nums, int numsSize) {
4     int* prefixSum = (int*)malloc((2 * numsSize + 1) * sizeof(int));
5     int maxLength = 0;
6     for (int i = 0; i < 2 * numsSize + 1; i++) {
7         prefixSum[i] = -2;
8     }
9     int sum = 0;
10    int index;
11    for (int i = 0; i < numsSize; i++) {
12        sum += (nums[i] == 0) ? -1 : 1;
13        if (sum == 0) {
14            maxLength = i + 1;
15        } else {
16            index = sum + numsSize;
17            if (prefixSum[index] == -2) {
18                prefixSum[index] = i;
19            } else {
20                maxLength = (i - prefixSum[index] > maxLength) ? i -
                    prefixSum[index] : maxLength;
21            }
22        }
23    }
24    free(prefixSum);
25    return maxLength;
}
```

Output

/tmp/CuWz9Av5iz.o  
The length of the longest subarray with equal number of 0s and 1s is: 8

### 4.deletion of duplicate

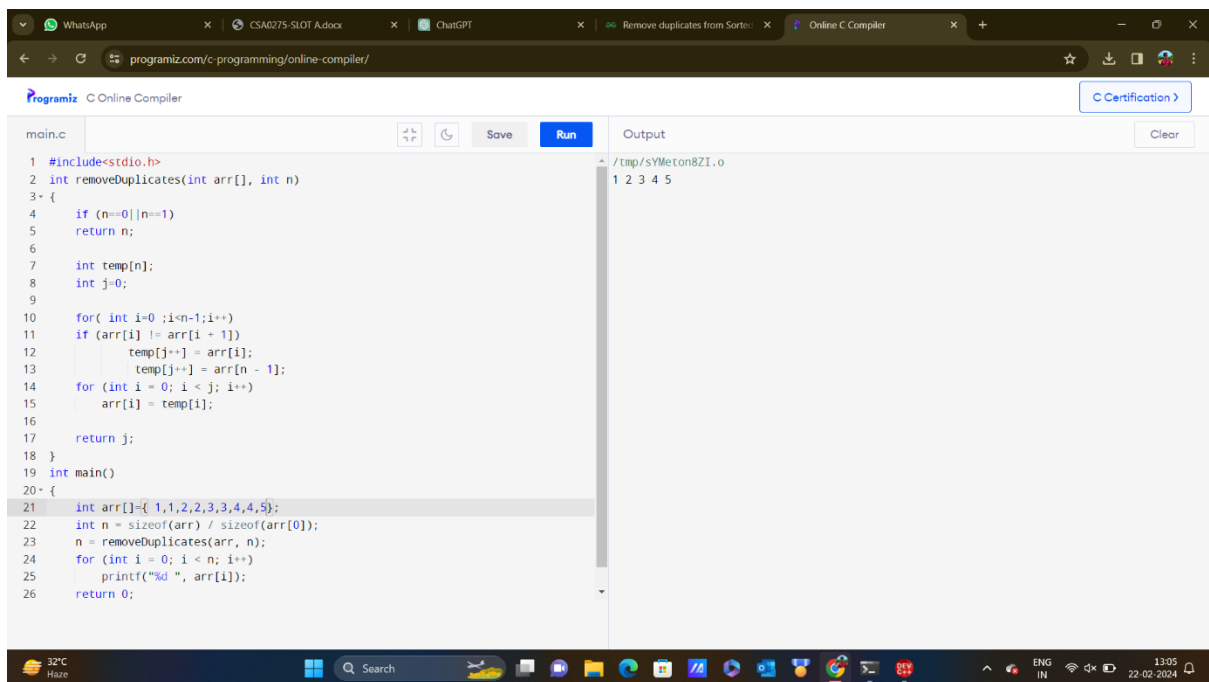


```
largeststring.cpp
1 #include<stdio.h>
2 #include<string.h>
3
4 int max(int a,int b)
5 {
6     return (a > b) ? a:b;
7 }
8
9 void longestSubstring(char *str)
10 {
11     int n=strlen(str),maxLength = 0,start = 0,startOfMax = 0;
12     int lastOccurrence[256];
13     memset(lastOccurrence, -1, sizeof(lastOccurrence));
14
15     for (int end = 0; end < n; end++)
16     {
17         start = max(start, lastOccurrence[str[end]] + 1);
18
19         if (maxLength < end - start + 1)
20         {
21             maxLength = end - start + 1;
22             startOfMax = start;
23         }
24         lastOccurrence[str[end]] = end;
25     }
26
27     printf("Longest Substring Without Repeating Characters: %s", str[startOfMax]);
28     for (int i = startOfMax; i < startOfMax + maxLength; i++)
29     {
30         printf("%c", str[i]);
31     }
32     printf("\n");
33 }
34
35 int main() {
36     char str[] = "saveetha";
37     longestSubstring(str);
38     return 0;
39 }
```

rs: "save"

Process exited after 10.08 seconds with return value 0  
Press any key to continue . . .

## 5.non repeated elements in an array

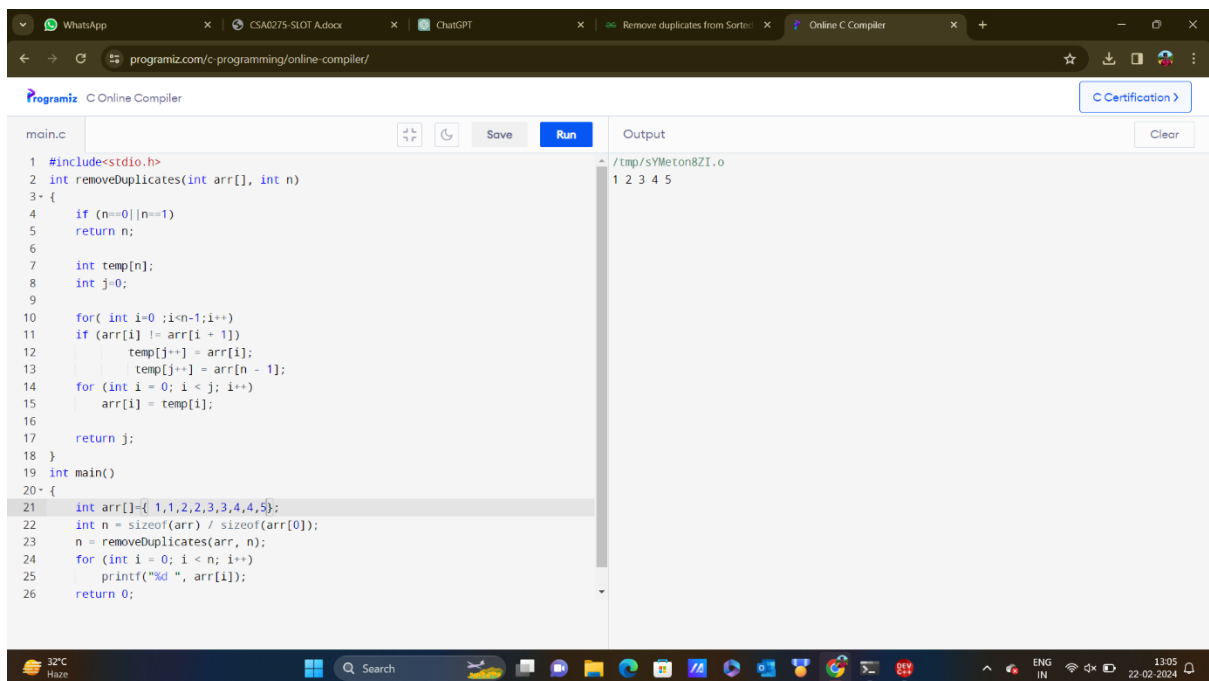


The screenshot shows a web browser with multiple tabs, including 'Online C Compiler'. The compiler interface displays a C program in a text editor on the left and its output on the right. The program is designed to remove duplicate elements from an array. The output shows the array after duplicates have been removed, resulting in the sequence 1 2 3 4 5.

```
main.c
1 #include<stdio.h>
2 int removeDuplicates(int arr[], int n)
3 {
4     if (n==0||n==1)
5         return n;
6
7     int temp[n];
8     int j=0;
9
10    for( int i=0 ;i<n-1;i++)
11        if (arr[i] != arr[i + 1])
12            temp[j++] = arr[i];
13        temp[j++] = arr[n - 1];
14    for (int i = 0; i < j; i++)
15        arr[i] = temp[i];
16
17    return j;
18 }
19 int main()
20 {
21     int arr[]={1,1,2,2,3,3,4,4,5};
22     int n = sizeof(arr) / sizeof(arr[0]);
23     n = removeDuplicates(arr, n);
24     for (int i = 0; i < n; i++)
25         printf("%d ", arr[i]);
26     return 0;
}
```

Output: /tmp/sYMeTon8ZI.o  
1 2 3 4 5

## 6.

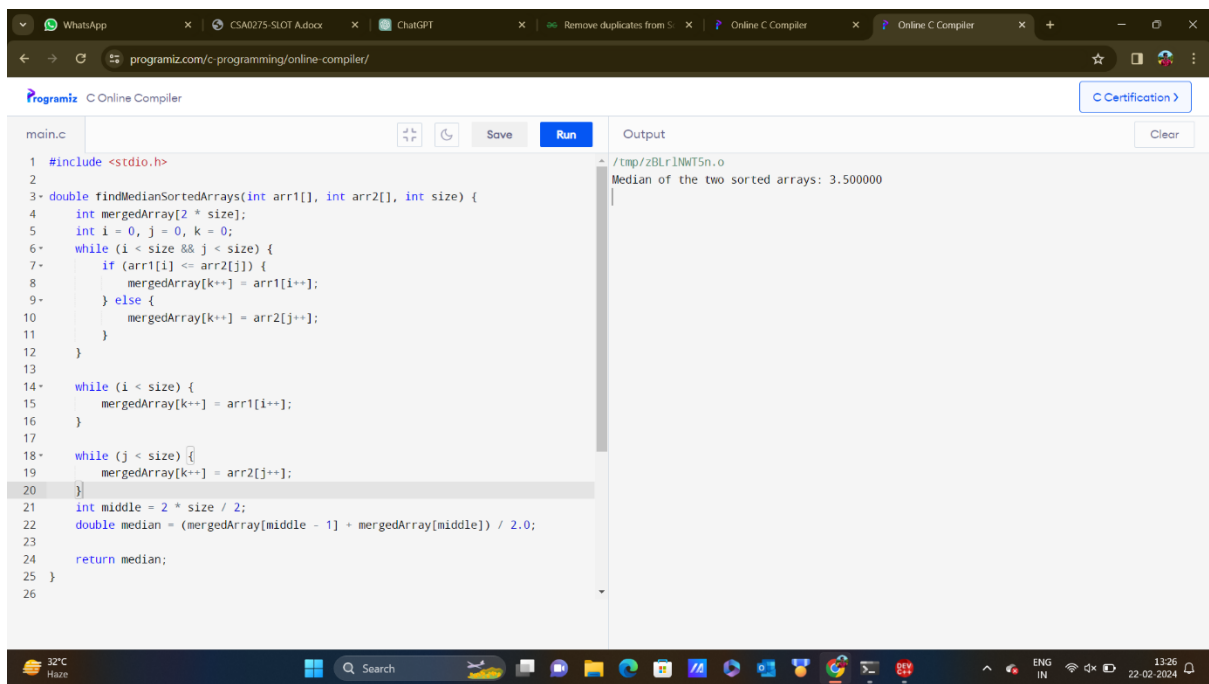


This screenshot is identical to the one above, showing the same C program and its output. The program removes duplicate elements from the array {1,1,2,2,3,3,4,4,5}, resulting in the output 1 2 3 4 5.

```
main.c
1 #include<stdio.h>
2 int removeDuplicates(int arr[], int n)
3 {
4     if (n==0||n==1)
5         return n;
6
7     int temp[n];
8     int j=0;
9
10    for( int i=0 ;i<n-1;i++)
11        if (arr[i] != arr[i + 1])
12            temp[j++] = arr[i];
13        temp[j++] = arr[n - 1];
14    for (int i = 0; i < j; i++)
15        arr[i] = temp[i];
16
17    return j;
18 }
19 int main()
20 {
21     int arr[]={1,1,2,2,3,3,4,4,5};
22     int n = sizeof(arr) / sizeof(arr[0]);
23     n = removeDuplicates(arr, n);
24     for (int i = 0; i < n; i++)
25         printf("%d ", arr[i]);
26     return 0;
}
```

Output: /tmp/sYMeTon8ZI.o  
1 2 3 4 5

## 7. median of given array.



The screenshot shows a web browser with multiple tabs, including WhatsApp, CSA0275 SLOT Ad doc, ChatGPT, and Online C Compiler. The active tab is the Online C Compiler, which displays a C program in a text editor and its output in a separate pane.

**Programiz C Online Compiler**

**main.c**

```
1 #include <stdio.h>
2
3 double findMedianSortedArrays(int arr1[], int arr2[], int size) {
4     int mergedArray[2 * size];
5     int i = 0, j = 0, k = 0;
6     while (i < size && j < size) {
7         if (arr1[i] <= arr2[j]) {
8             mergedArray[k++] = arr1[i++];
9         } else {
10            mergedArray[k++] = arr2[j++];
11        }
12    }
13
14    while (i < size) {
15        mergedArray[k++] = arr1[i++];
16    }
17
18    while (j < size) {
19        mergedArray[k++] = arr2[j++];
20    }
21
22    int middle = 2 * size / 2;
23    double median = (mergedArray[middle - 1] + mergedArray[middle]) / 2.0;
24
25    return median;
26 }
```

**Output**

```
/tmp/zBLr1NWt5n.o
Median of the two sorted arrays: 3.500000
```

The Windows taskbar at the bottom shows the system clock as 13:26 on 22-02-2024, with a temperature of 32°C and weather conditions of Haze.