

main.c



Save

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #define NO_OF_CHARS 256
5 int max(int a, int b) {
6     return (a > b) ? a : b;
7 }
8 int longestUniqueSubsttr(char *str) {
9     int n = strlen(str);
10    int cur_len = 1;
11    int max_len = 1;
12    int prev_index;
13    int i;
14    int *visited = (int *)malloc(sizeof(int) * NO_OF_CHARS);
15    for (i = 0; i < NO_OF_CHARS; i++)
16        visited[i] = -1;
17    visited[str[0]] = 0;
18    for (i = 1; i < n; i++) {
19        prev_index = visited[str[i]];
20        if (prev_index == -1 || i - cur_len > prev_index)
```

```
/tmp/S94r2uVm96.o
The input string is: abcabcbb
The length of the longest non-repeating substring is: 3
```

main.c



Save

Run

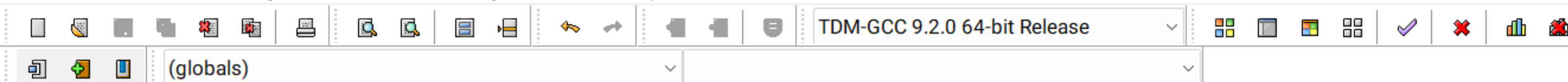
Output

Clear

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 char* longestCommonPrefix(char* arr[], int n) {
5     if (n == 0) return "";
6     if (n == 1) return arr[0];
7     qsort(arr, n, sizeof(char*), strcmp);
8     int len = strlen(arr[0]);
9     char* first = arr[0];
10    char* last = arr[n - 1];
11    int i = 0;
12    while (i < len && first[i] == last[i]) {
13        i++;
14    }
15    char* prefix = (char*)malloc(i + 1);
16    strncpy(prefix, first, i);
17    prefix[i] = '\0';
18
19    return prefix;
20 }
```

/tmp/0qAU9Un0fd.o

The longest common prefix is: fl



Project Class max product.cpp

```
13
14     int maxEndingHere = arr[0];
15     int minEndingHere = arr[0];
16     int maxSoFar = arr[0];
17
18     for (int i = 1; i < n; i++) {
19         if (arr[i] < 0) {
20             int temp = maxEndingHere;
21             maxEndingHere = max(arr[i], arr[i] * minEndingHere);
22             minEndingHere = min(arr[i], arr[i] * temp);
23         } else {
24             maxEndingHere = max(arr[i], arr[i] * maxEndingHere);
25             minEndingHere = min(arr[i], arr[i] * minEndingHere);
26         }
27
28         maxSoFar = max(maxSoFar, maxEndingHere);
29     }
30
31     return maxSoFar;
32 }
33
34 int main() {
35     int arr[] = {2, 3, -2, 4};
36     int n = sizeof(arr) / sizeof(arr[0]);
37     int maxProduct = maxProductSubarray(arr, n);
38     printf("The maximum product of a subarray is %d\n", maxProduct);
39     return 0;
40 }
41
```

Compiler Resources Compile Log Debug Find Results Console Close

Abort Compilation

☐ Shorten compiler path

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Byalla Vishnu\Desktop\max product.exe
- Output Size: 323.6923828125 KiB
- Compilation Time: 0.28s
```

Line: 27 Col: 1 Sel: 0 Lines: 41 Length: 1005 Insert Done parsing in 0 seconds

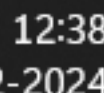
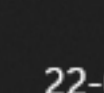
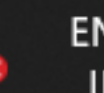
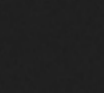
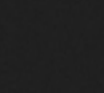
C:\Users\Byalla Vishnu\Desktop\max product.exe

The maximum product of a subarray is 6

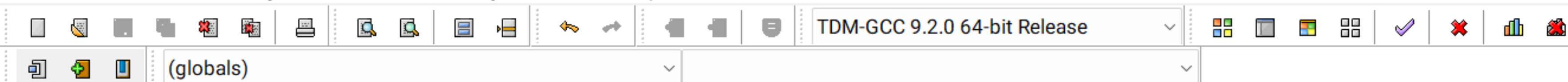
```
-----
Process exited after 3.018 seconds with return value 0
Press any key to continue . . .
```



Search







Project Class max product.cpp x remove duplicate elements.c x

```
1 #include <stdio.h>
2 int removeDuplicates(int arr[], int n) {
3     if (n == 0 || n == 1) return n;
4     int j = 0;
5     for (int i = 0; i < n - 1; i++) {
6         if (arr[i] != arr[i + 1]) {
7             arr[j++] = arr[i];
8         }
9     }
10    arr[j++] = arr[n - 1];
11    return j;
12 }
13 int main() {
14     int arr[] = {1, 1, 2, 2, 3, 4, 4, 5, 6, 6};
15     int n = sizeof(arr) / sizeof(arr[0]);
16     printf("Original sorted array:\n");
17     for (int i = 0; i < n; i++) {
18         printf("%d ", arr[i]);
19     }
20     printf("\n");
21     n = removeDuplicates(arr, n);
22     printf("Array after removing duplicates:\n");
23     for (int i = 0; i < n; i++) {
24         printf("%d ", arr[i]);
25     }
26     printf("\n");
27     return 0;
28 }
```

Compiler Resources Compile Log Debug Find Results Console Close

Abort Compilation

☐ Shorten compiler path

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Byalla Vishnu\Desktop\remove duplicate elements.c
- Output Size: 323.5166015625 KiB
- Compilation Time: 0.33s
```

Line: 11 Col: 15 Sel: 0 Lines: 29 Length: 729 Insert Done parsing in 0 seconds

C:\Users\Byalla Vishnu\Desktop

Original sorted array:

1 1 2 2 3 4 4 5 6 6

Array after removing duplicates:

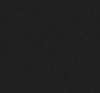
1 2 3 4 5 6

-----  
Process exited after 3.484 seconds with return value 0

Press any key to continue . . .



Search

ENG  
IN13:43  
22-02-2024

```
1  #include <stdio.h>
2
3  // Function to find the index of the first occurrence of a
   target value in a sorted array
4  int binarySearchFirstOccurrence(int arr[], int size, int
   target) {
5      int low = 0, high = size - 1;
6      int result = -1; // Initialize result to -1 (not
   found)
7
8      while (low <= high) {
9          int mid = low + (high - low) / 2;
10
11         if (arr[mid] == target) {
12             // Update result and search towards the left
               for the first occurrence
13             result = mid;
14             high = mid - 1;
15         } else if (arr[mid] < target) {
16             // If target is greater, search in the right
               half
17             low = mid + 1;
18         } else {
19             // If target is smaller, search in the left
               half
20             high = mid - 1;
21         }
22     }
23
24     return result;
25 }
26
27 int main() {
28     int arr[] = {1, 2, 2, 4, 4, 4, 5, 6};
29     int size = sizeof(arr) / sizeof(arr[0]);
```

```
$ /tmp/a.out
The first occurrence of 4 is at index 3

```



```
1  #include <stdio.h>
2
3  #define MAX_SIZE 100
4
5
6  void generateCombinations(int arr[], int data[], int
   start, int end, int index, int k);
7
8  // Function to print an array
9  void printArray(int arr[], int size);
10
11 int main() {
12     int arr[] = {1, 2, 3, 4};
13     int n = sizeof(arr) / sizeof(arr[0]);
14     int k = 2; // Set the length of combinations
15
16     int data[k]; // Temporary array to store combinations
17
18     // Generate combinations
19     generateCombinations(arr, data, 0, n - 1, 0, k);
20
21     return 0;
22 }
23
24 void generateCombinations(int arr[], int data[], int
   start, int end, int index, int k) {
25     // If a combination of length k is found, print it
26     if (index == k) {
27         printArray(data, k);
28         return;
29     }
30
31     // Iterate over the array and generate combinations
32     for (int i = start; i <= end && end - i + 1 >= k -
```

/tmp/a.out

1 2

1 3

1 4

2 3

2 4

3 4

□



```
1  #include <stdio.h>
2  #include <stdbool.h>
3
4  #define SIZE 9
5
6  // Function to check if a Sudoku board is valid
7  bool isValidSudoku(char board[SIZE][SIZE]) {
8      // Check each row and column
9      for (int i = 0; i < SIZE; i++) {
10         // Arrays to keep track of seen digits in rows and
           columns
11         int rowSet[SIZE] = {0};
12         int colSet[SIZE] = {0};
13
14         for (int j = 0; j < SIZE; j++) {
15             // Check rows
16             if (board[i][j] != '.' && rowSet[board[i][j] -
               '1'] == 1) {
17                 return false; // Duplicate in the same row
18             }
19             rowSet[board[i][j] - '1'] = 1;
20
21             // Check columns
22             if (board[j][i] != '.' && colSet[board[j][i] -
               '1'] == 1) {
23                 return false; // Duplicate in the same
               column
24             }
25             colSet[board[j][i] - '1'] = 1;
26         }
27     }
28
29     // Check each 3x3 subgrid
30     for (int block = 0; block < SIZE; block += 3) {
31         for (int i = 0; i < SIZE; i += 3) {
```

```
$ /tmp/a.out
The Sudoku board is valid.
█
```



C Certification >

```
main.c
36 return 0;
37 }
38 void rotateByOne(int arr[])
39 {
40     int i, last;
41     last = arr[SIZE - 1];
42     for(i=SIZE-1; i>0; i--)
43     {
44         arr[i] = arr[i - 1];
45     }
46     arr[0] = last;
47 }
48
49
50 /**
51  * Print the given array
52  */
53 void printArray(int arr[])
54 {
55     int i;
56
57     for(i=0; i<SIZE; i++)
58     {
59         printf("%d ", arr[i]);
60     }
61 }
```

Save Run

Output

Clear

```
/tmp/wY977cN9KA.o
Enter 10 elements array: 1
2
3
4
5
6
7
8
9
2
Enter number of times to right rotate: 10
Array before rotationn1 2 3 4 5 6 7 8 9 2
Array after rotation
1 2 3 4 5 6 7 8 9 2
```

**C Course, Enhanced by AI**

Learn c the right way – solve challenges, build projects, and leverage the power of AI to aid you in handling errors.

[Get Started for Free](#)