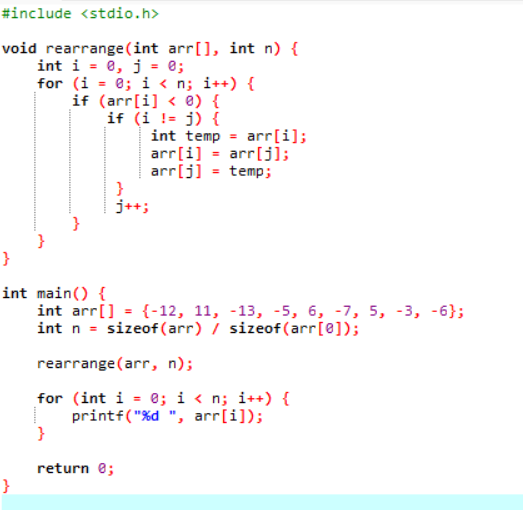
INTERMEDIATE LEVEL

Day 1:22.02.2024

1.give an array integer ,rearrange the elements in such a way that all negative elements come before the positive elements.

Code:



2.create a structural named “student” to store students details such as name ,roll number, and marks in three subjects .write a program to calculate the average marks of the students.

Code:

#include <stdio.h>

struct student {

char name[50];

int roll\_number;

int marks[3];

};

float calculateAverage(struct student s) {

float sum = 0;

for (int i = 0; i < 3; i++) {

sum += s.marks[i];

}

return sum / 3.0;

}

int main() {

struct student s;

printf("Enter student details:\n");

printf("Name: ");

scanf("%s", s.name);

printf("Roll Number: ");

scanf("%d", &s.roll\_number);

printf("Enter marks in three subjects: ");

for (int i = 0; i < 3; i++) {

scanf("%d", &s.marks[i]);

}

printf("\nStudent Details:\n");

printf("Name: %s\n", s.name);

printf("Roll Number: %d\n", s.roll\_number);

printf("Marks in three subjects: %d, %d, %d\n", s.marks[0], s.marks[1], s.marks[2]);

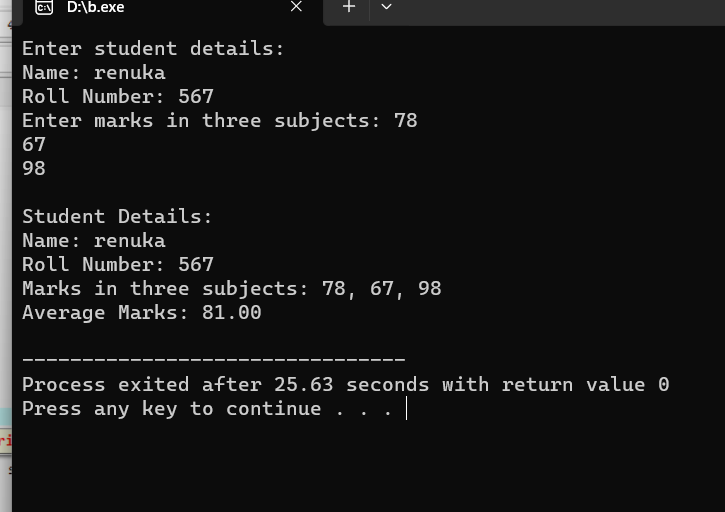
float average = calculateAverage(s);

printf("Average Marks: %.2f\n", average);

return 0;

}

Output:



3.define a structured name “date” to represent a date .

Code:

#include <stdio.h>

struct date {

int day;

int month;

int year;

};

int compareDates(struct date date1, struct date date2) {

if (date1.year < date2.year) {

return 1;

} else if (date1.year > date2.year) {

return 2;

} else {

if (date1.month < date2.month) {

return 1;

} else if (date1.month > date2.month) {

return 2;

} else {

if (date1.day < date2.day) {

return 1;

} else if (date1.day > date2.day) {

return 2;

} else {

return 0;

}

}

}

}

int main() {

struct date date1, date2;

printf("Enter the first date (dd mm yyyy): ");

scanf("%d %d %d", &date1.day, &date1.month, &date1.year);

printf("Enter the second date (dd mm yyyy): ");

scanf("%d %d %d", &date2.day, &date2.month, &date2.year);

int result = compareDates(date1, date2);

if (result == 0) {

printf("Both dates are the same.\n");

} else if (result == 1) {

printf("First date comes first.\n");

} else {

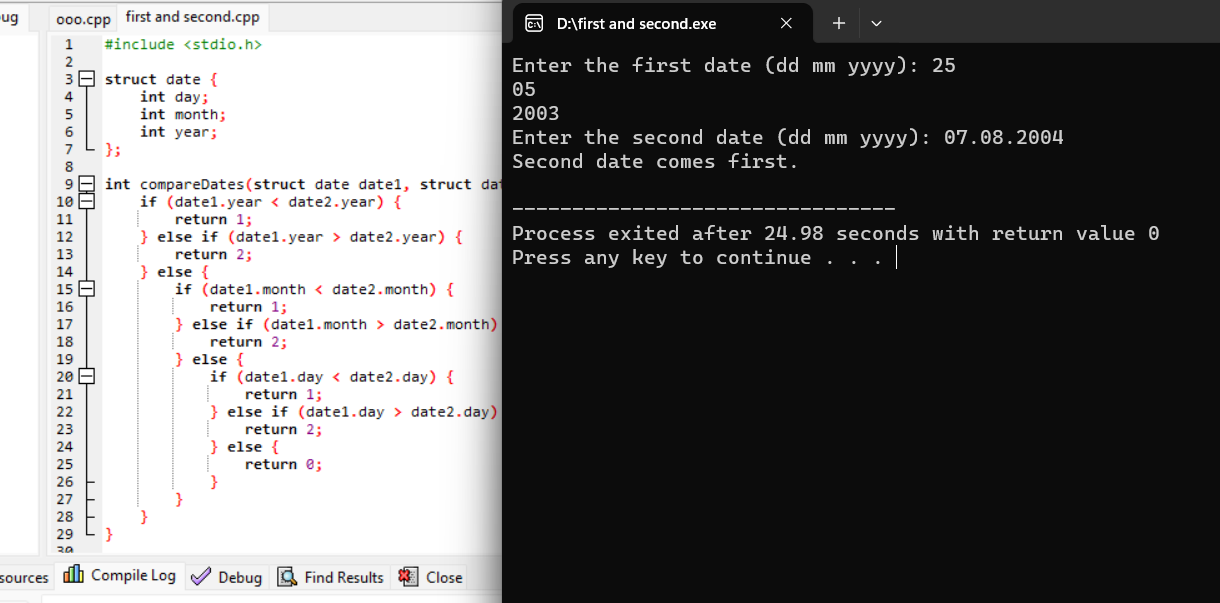
printf("Second date comes first.\n");

}

return 0;

}

Output:



4. Implement a recursive program to solve the Sudoku puzzle

Code:

#include <stdio.h>

#define N 9

void printGrid(int grid[N][N]) {

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

for (int j = 0; j < N; j++) {

printf("%2d", grid[i][j]);

}

printf("\n");

}

}

int isSafe(int grid[N][N], int i, int j, int num) {

for (int x = 0; x < N; x++) {

if (grid[i][x] == num || grid[x][j] == num) {

return 0;

}

}

int startRow = i - i % 3;

int startCol = j - j % 3;

for (int row = 0; row < 3; row++) {

for (int col = 0; col < 3; col++) {

if (grid[row + startRow][col + startCol] == num) {

return 0;

}

}

}

return 1;

}

int solveSudoku(int grid[N][N], int i, int j) {

if (i == N - 1 && j == N) {

return 1;

}

if (j == N) {

i++;

j = 0;

}

if (grid[i][j] != 0) {

return solveSudoku(grid, i, j + 1);

}

for (int num = 1; num <= N; num++) {

if (isSafe(grid, i, j, num)) {

grid[i][j] = num;

if (solveSudoku(grid, i, j + 1)) {

return 1;

}

grid[i][j] = 0;

}

}

return 0;

}

int main() {

int grid[N][N] = {

{5, 3, 0, 0, 7, 0, 0, 0, 0},

{6, 0, 0, 1, 9, 5, 0, 0, 0},

{0, 9, 8, 0, 0, 0, 0, 6, 0},

{8, 0, 0, 0, 6, 0, 0, 0, 3},

{4, 0, 0, 8, 0, 3, 0, 0, 1},

{7, 0, 0, 0, 2, 0, 0, 0, 6},

{0, 6, 0, 0, 0, 0, 2, 8, 0},

{0, 0, 0, 4, 1, 9, 0, 0, 5},

{0, 0, 0, 0, 8, 0, 0, 7, 9}

};

if (solveSudoku(grid, 0, 0)) {

printf("Sudoku solution:\n");

printGrid(grid);

} else {

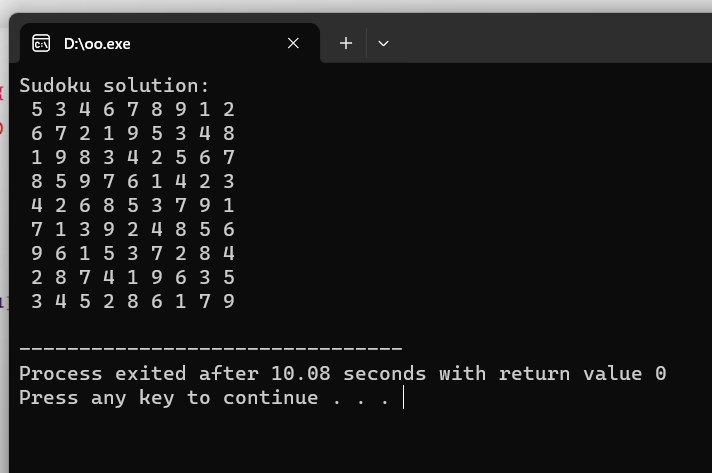
printf("No solution exists.\n");

}

return 0;

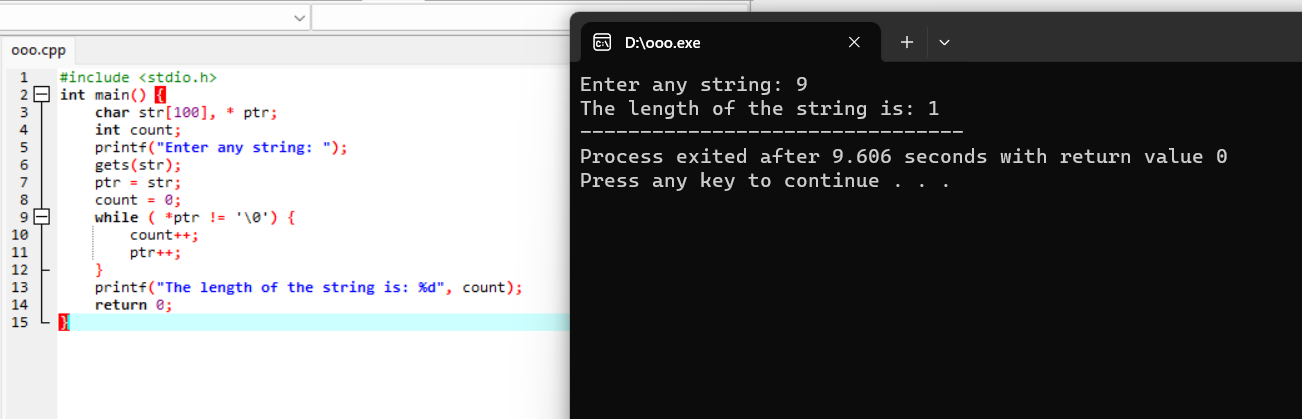
}

Output:



5. Write a program to find the length of a string using pointers.

CODE:



6. Write a C program to find the longest common prefix string amongst an array of strings.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int min(int a, int b) {

return (a < b) ? a : b;

}

char\* longestCommonPrefix(char\*\* strs, int strsSize) {

if (strsSize == 0)

return "";

int minLen = strlen(strs[0]);

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

minLen = min(minLen, strlen(strs[i]));

int i, j;

for (i = 0; i < minLen; ++i) {

for (j = 1; j < strsSize; ++j) {

if (strs[j][i] != strs[j - 1][i])

break;

}

if (j != strsSize)

break;

}

char\* commonPrefix = (char\*)malloc(sizeof(char) \* (i + 1));

strncpy(commonPrefix, strs[0], i);

commonPrefix[i] = '\0';

return commonPrefix;

}

int main() {

char \*strings[] = {"flower", "flow", "flight"};

int size = sizeof(strings) / sizeof(strings[0]);

char \*prefix = longestCommonPrefix(strings, size);

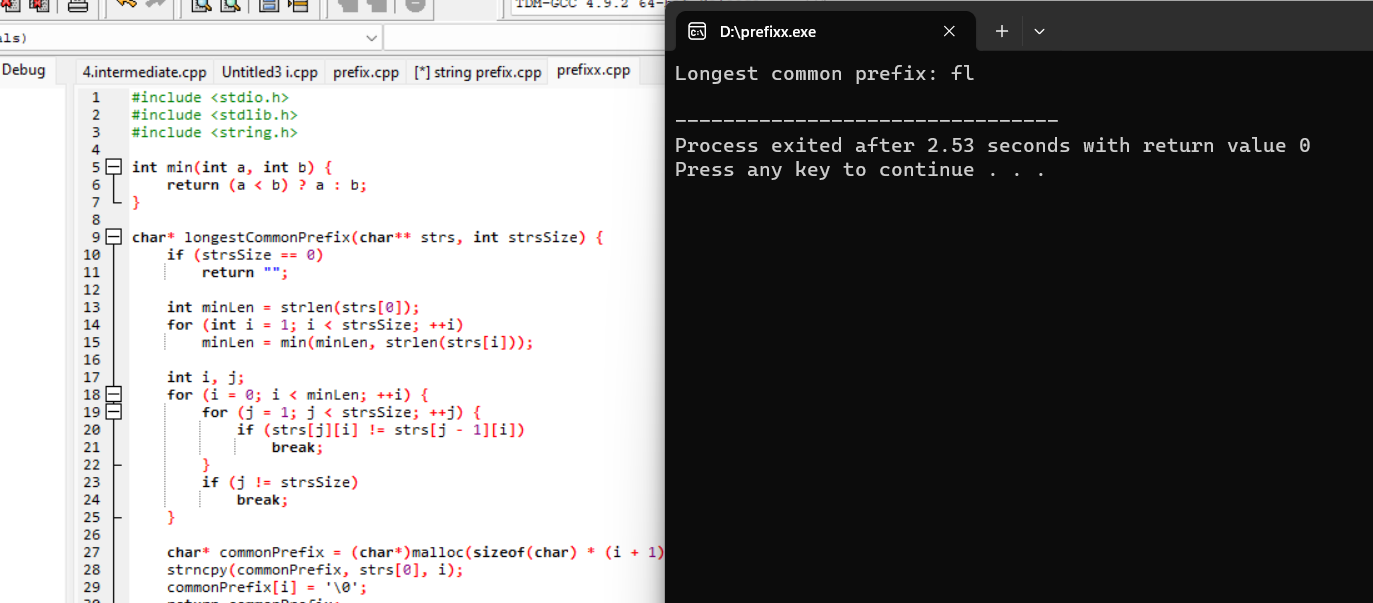
printf("Longest common prefix: %s\n", prefix);

free(prefix);

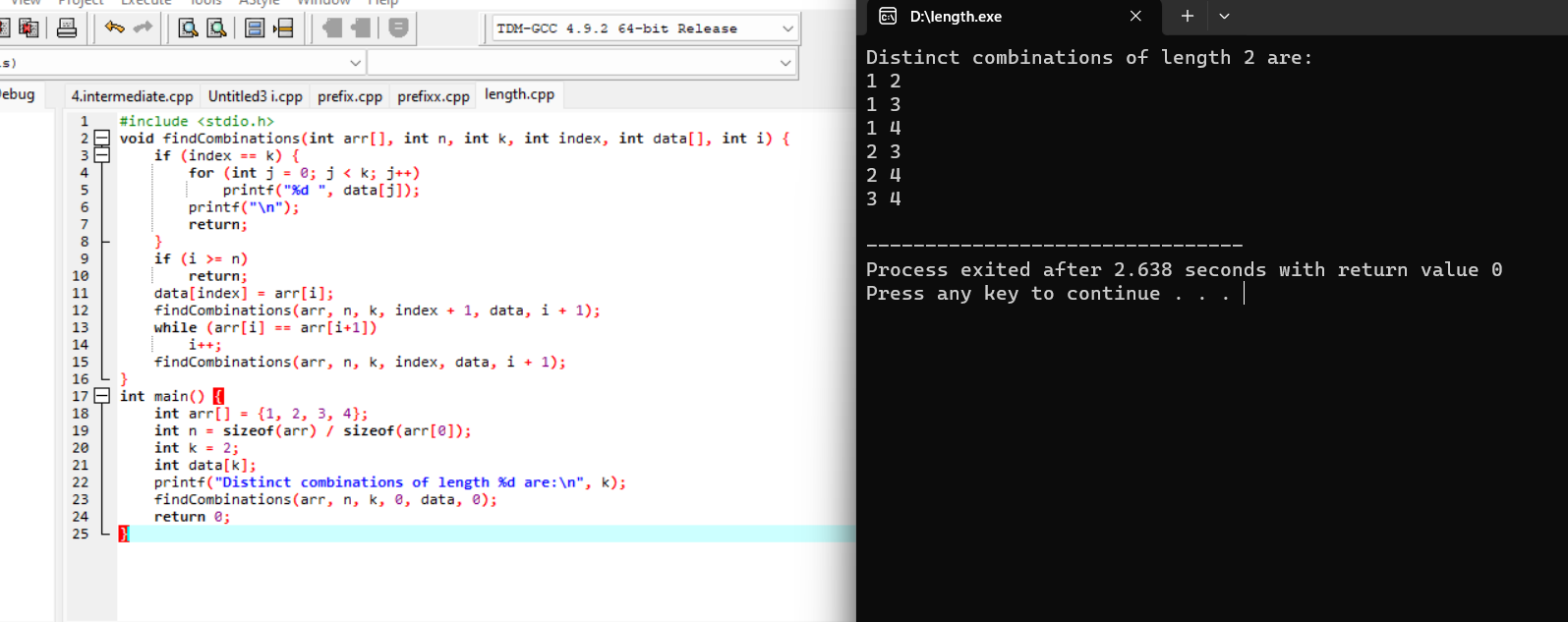
return 0;

}

Outcome:



7. Write a C program to implement a function that rotates a given array of integers to the right by k steps



8. Write a C program to implement a function that returns the index of the first occurrence of a target value in a sorted array.

#include <stdio.h>

int findFirstOccurrence(int arr[], int size, int target) {

int low = 0;

int high = size - 1;

int result = -1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target) {

result = mid;

high = mid - 1;

} else if (arr[mid] < target) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return result;

}

int main() {

int arr[] = {1, 2, 3, 4, 4, 4, 5, 6};

int size = sizeof(arr) / sizeof(arr[0]);

int target = 3;

int index = findFirstOccurrence(arr, size, target);

if (index != -1) {

printf("First occurrence of %d is at index %d\n", target, index);

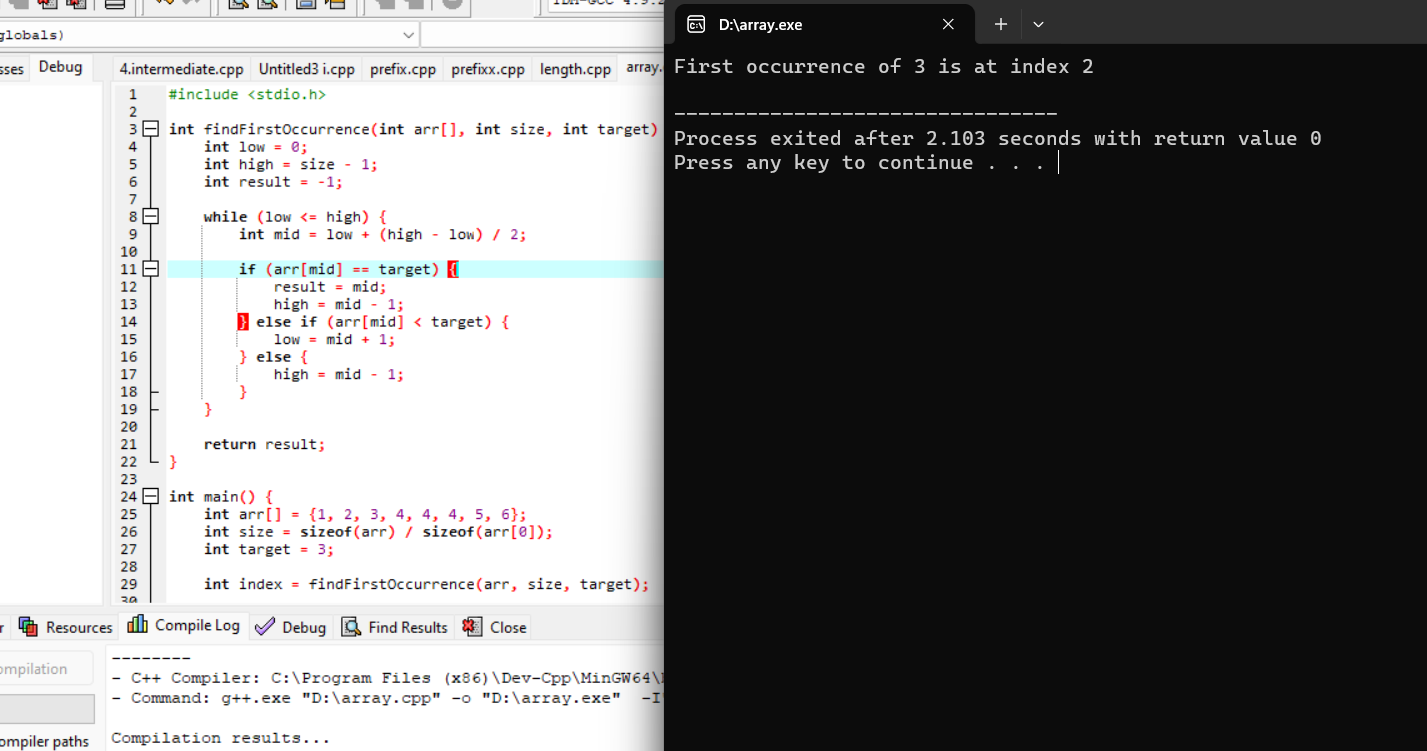
} else {

printf("%d not found in the array\n", target);

}

return 0;

}



9.Write a C program to find the longest substring without repeating characters in a given string.

#include <stdio.h>

#include <string.h>

#define MAX\_CHARS 256

int max(int a, int b) {

return (a > b) ? a : b;

}

void longestUniqueSubsttr(char \*str) {

int n = strlen(str);

int visited[MAX\_CHARS];

int start = 0;

int currentLength = 1;

int maxLength = 1;

int previousIndex;

for (int i = 0; i < MAX\_CHARS; i++)

visited[i] = -1;

visited[str[0]] = 0;

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

previousIndex = visited[str[i]];

if (previousIndex == -1 || i - currentLength > previousIndex)

currentLength++;

else {

if (currentLength > maxLength) {

maxLength = currentLength;

start = i - maxLength;

}

currentLength = i - previousIndex;

}

visited[str[i]] = i;

}

if (currentLength > maxLength) {

maxLength = currentLength;

start = n - maxLength;

}

printf("The longest substring without repeating characters is: ");

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

printf("%c", str[i]);

printf("\n");

}

int main() {

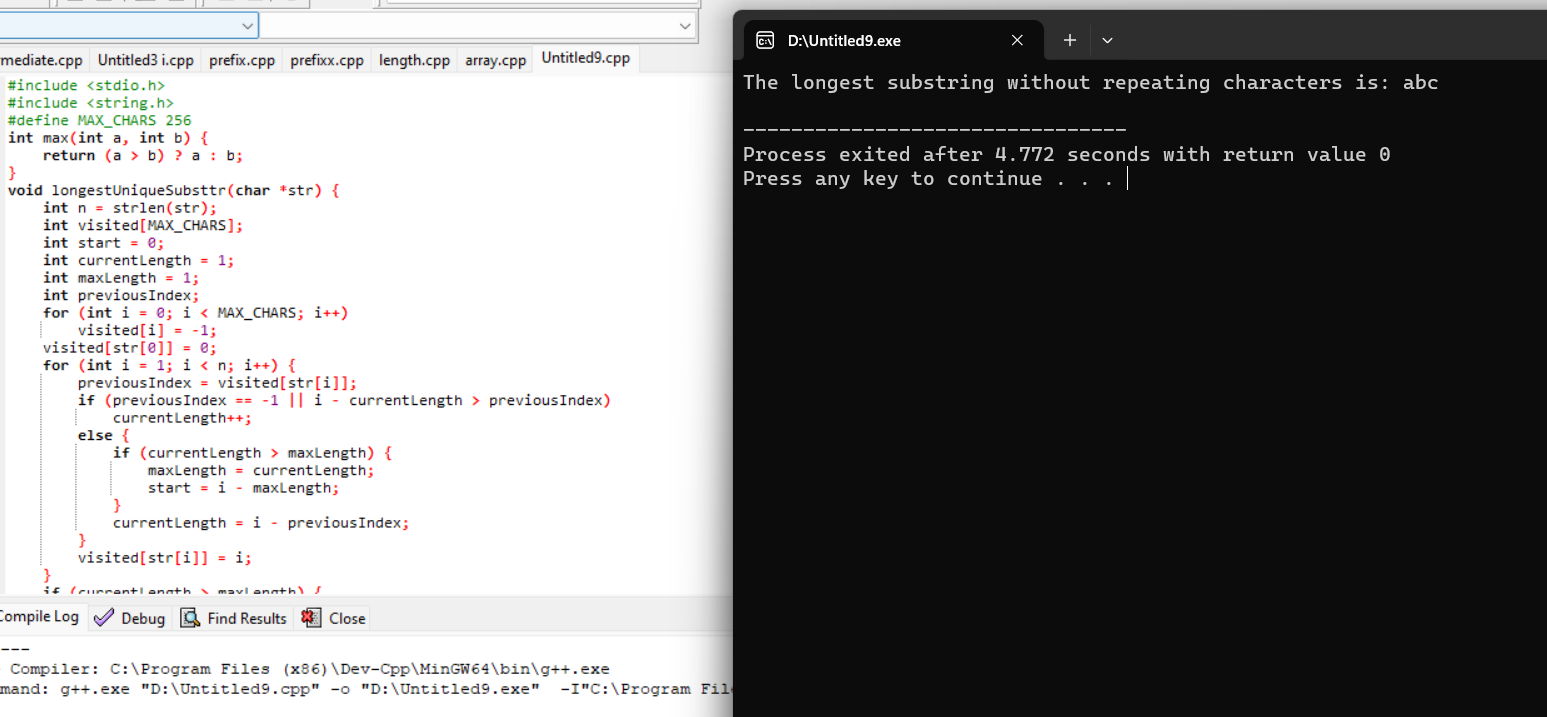
char str[] = "abcabcbb";

longestUniqueSubsttr(str);

return 0;

}

Output:



10.Write a C program to find the maximum product subarray in a given array of integers.

#include <stdio.h>

int max(int a, int b) {

return (a > b) ? a : b;

}

int min(int a, int b) {

return (a < b) ? a : b;

}

int maxProductSubarray(int arr[], int n) {

int maxEndingHere = 1;

int minEndingHere = 1;

int maxProduct = 1;

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

if (arr[i] > 0) {

maxEndingHere = maxEndingHere \* arr[i];

minEndingHere = min(minEndingHere \* arr[i], 1);

}

else if (arr[i] == 0) {

maxEndingHere = 1;

minEndingHere = 1;

}

else {

int temp = maxEndingHere;

maxEndingHere = max(minEndingHere \* arr[i], 1);

minEndingHere = temp \* arr[i];

}

if (maxProduct < maxEndingHere) {

maxProduct = maxEndingHere;

}

}

return maxProduct;

}

int main() {

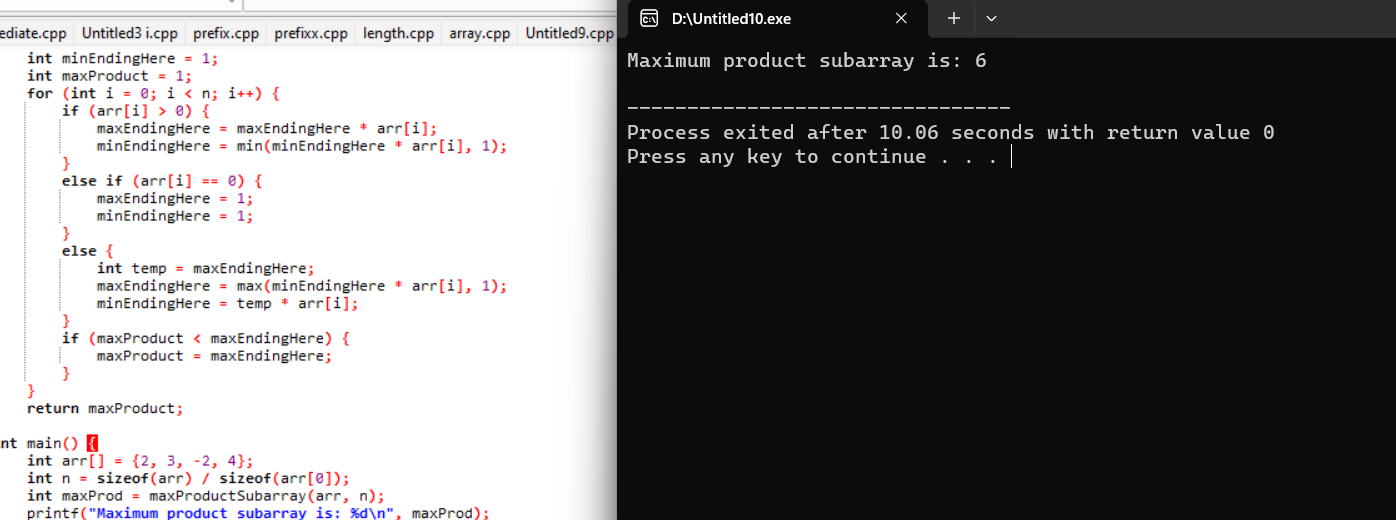
int arr[] = {2, 3, -2, 4};

int n = sizeof(arr) / sizeof(arr[0]);

int maxProd = maxProductSubarray(arr, n);

printf("Maximum product subarray is: %d\n", maxProd);

return 0;

}11.Write a C program to find all distinct combinations of a given length k from a given array of integers without using the same element twice.

Code:

#include <stdio.h>

void findCombinations(int arr[], int n, int k, int index, int data[], int i) {

if (index == k) {

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

printf("%d ", data[j]);

printf("\n");

return;

}

if (i >= n)

return;

data[index] = arr[i];

findCombinations(arr, n, k, index + 1, data, i + 1);

while (arr[i] == arr[i+1])

i++;

findCombinations(arr, n, k, index, data, i + 1);

}

int main() {

int arr[] = {1, 2, 3, 4};

int n = sizeof(arr) / sizeof(arr[0]);

int k = 2;

int data[k];

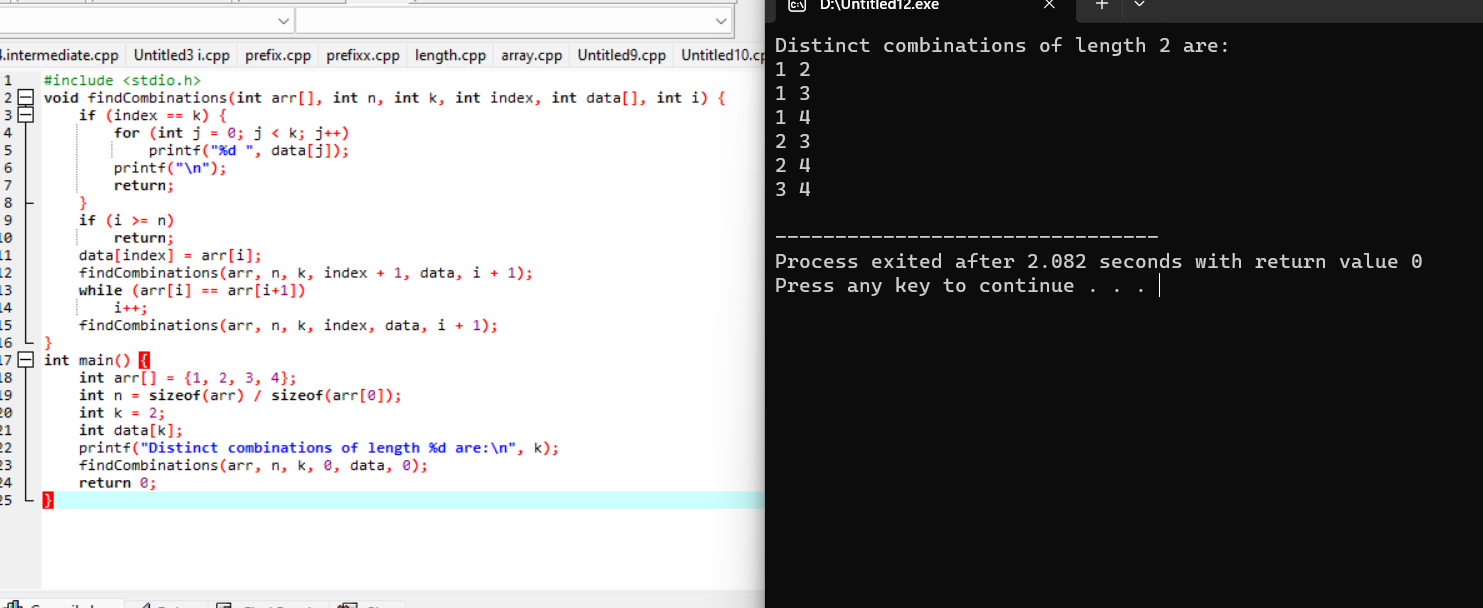
printf("Distinct combinations of length %d are:\n", k);

findCombinations(arr, n, k, 0, data, 0);

return 0;

}

Output:



12. Write a C program to implement a function that returns the minimum number of jumps needed to reach the end of an array, where each element represents the maximum jump length

Code:

#include <stdio.h>

#include <limits.h>

int minJumps(int arr[], int n) {

if (n <= 1)

return 0;

if (arr[0] == 0)

return -1;

int maxReach = arr[0];

int steps = arr[0];

int jumps = 1;

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

if (i == n - 1)

return jumps;

maxReach = (maxReach > i + arr[i]) ? maxReach : (i + arr[i]);

steps--;

if (steps == 0) {

jumps++;

if (i >= maxReach)

return -1;

steps = maxReach - i;

}

}

return -1;

}

int main() {

int arr[] = {2, 3, 1, 1, 2, 4, 2, 0, 1, 1};

int n = sizeof(arr) / sizeof(arr[0]);

int result = minJumps(arr, n);

if (result != -1)

printf("Minimum number of jumps needed: %d\n", result);

else

printf("It is not possible to reach the end of the array.\n");

return 0;

}