

**ASSIGNMENT-I**

**1)Write a program in C to convert miles into kilometres (Km). Hint: 1 Mile=1.609 Km. [Use macros, relevant name and types for variables].**

**Sol:-**

#include <stdio.h>

#define MILE\_TO\_KM 1.609

int main() {

double miles, kilometres;

printf("Enter distance in miles: ");

scanf("%lf", &miles);

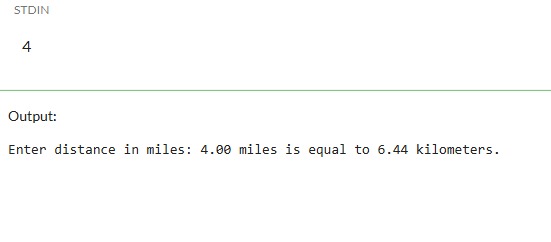
kilometers = miles \* MILE\_TO\_KM;

printf("%.2lf miles is equal to %.2lf kilometers.\n", miles, kilometers);

return 0;

}

**Output: -**



**2) Write a program to find the number of positive, negative and zeros in a sequence of inputs (numbers) entered as data.**

Sol: -

#include <stdio.h>

int main() {

int number, count, positive = 0, negative = 0, zeros = 0;

printf("How many numbers do you want to enter? ");

scanf("%d", &count);

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

printf("Enter number %d: ", i);

scanf("%d", &number);

if (number > 0)

positive++;

else if (number < 0)

negative++;

else

zeros++;

}

printf("\nResults:\n");

printf("Positive numbers: %d\n", positive);

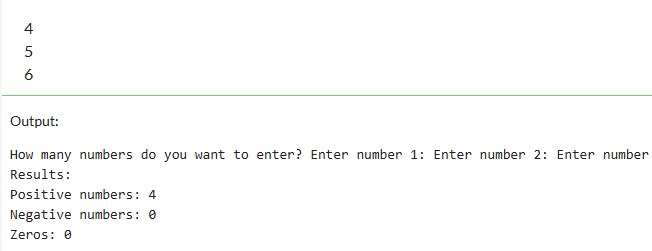
printf("Negative numbers: %d\n", negative);

printf("Zeros: %d\n", zeros);

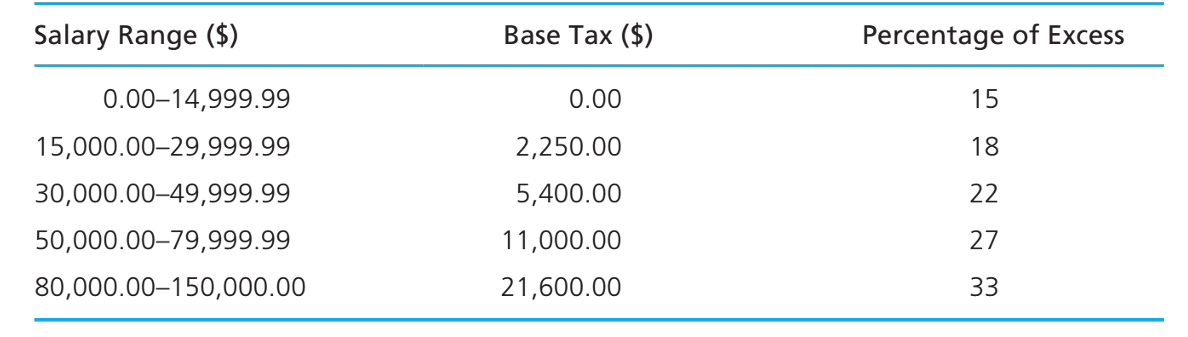
    return 0;

}

**Output: -**



**3) Compute the tax due based on a tax table given below: Program Input: Salary amount. Program Output: Returns the tax due for 0.0 <= salary <= 150,000.00; returns -1.0 if salary is outside the table range.**

****

**Sol: -**

#include <stdio.h>

double computeTax(double salary) {

double tax = 0.0;

if (salary < 0.0 || salary > 150000.0) {

return -1.0;

}

if (salary <= 14999.99) {

tax = 0.0;

} else if (salary <= 29999.99) {

tax = 2250.0 + (salary - 15000.0) \* 0.18;

} else if (salary <= 49999.99) {

tax = 5400.0 + (salary - 30000.0) \* 0.22;

} else if (salary <= 79999.99) {

tax = 11000.0 + (salary - 50000.0) \* 0.27;

} else {

tax = 21600.0 + (salary - 80000.0) \* 0.33;

}

return tax;

}

int main() {

double salary, tax;

printf("Enter the salary amount: ");

scanf("%lf", &salary);

tax = computeTax(salary);

if (tax == -1.0) {

printf("Salary is outside the table range.\n");

} else {

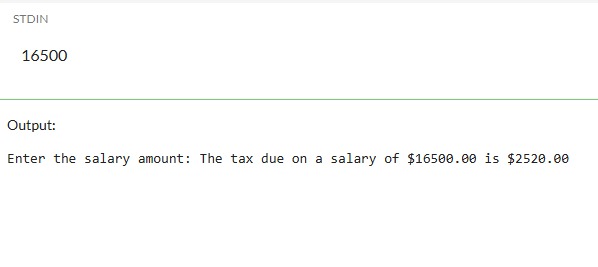
printf("The tax due on a salary of $%.2lf is $%.2lf\n", salary, tax);

}

return0;

}

**Output: -**



**4) Write an interactive program (menu driven) in ‘C’ (using functions) to compute the area of a selected geometrical figure from a list of such figures (square, rectangle, and circle).**

**Sol: -**

#include <stdio.h>

#include <stdlib.h>

#define PI 3.14159

float areaSquare(float side);

float areaRectangle(float length, float breadth);

float areaCircle(float radius);

int main() {

int choice;

float side, length, breadth, radius;

do {

printf("\n=== AREA CALCULATOR ===\n");

printf("1. Area of Square\n");

printf("2. Area of Rectangle\n");

printf("3. Area of Circle\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter side of square: ");

scanf("%f", &side);

printf("Area of square = %.2f\n", areaSquare(side));

break;

case 2:

printf("Enter length and breadth of rectangle: ");

scanf("%f %f", &length, &breadth);

printf("Area of rectangle = %.2f\n", areaRectangle(length, breadth));

break;

case 3:

printf("Enter radius of circle: ");

scanf("%f", &radius);

printf("Area of circle = %.2f\n", areaCircle(radius));

break;

case 4:

printf("Exiting program. Goodbye!\n");

exit(0);

default:

printf("Invalid choice! Please try again.\n");

}

} while (choice != 4);

return 0;

}

float areaSquare(float side) {

return side \* side;

}

float areaRectangle(float length, float breadth) {

return length \* breadth;

}

float areaCircle(float radius) {

return PI \* radius \* radius;

}

**Output: -**

****

**5) Write a program to display first n elements of Fibonacci series.**

**Sol: -**

#include <stdio.h>

int main() {

int n, i;

long long first = 0, second = 1, next;

printf("Enter the number of Fibonacci terms to display: ");

scanf("%d", &n);

if (n <= 0) {

printf("Please enter a positive integer.\n");

return 1;

}

printf("Fibonacci Series (%d terms):\n", n);

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

printf("%lld ", first);

next = first + second;

first = second;

second = next;

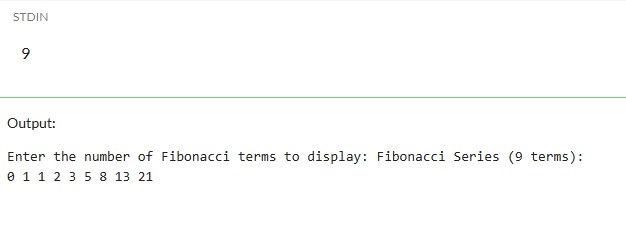
}

printf("\n");

    return 0;

}

**Output: -**



**6) Write a program to print a table book from Table X to Table Y. X and Y are user inputs.**

**Sol: -**

#include <stdio.h>

int main() {

int startTable, endTable, i, j;

printf("Enter the starting table (X): ");

scanf("%d", &startTable);

printf("Enter the ending table (Y): ");

scanf("%d", &endTable);

if (startTable > endTable) {

printf("Invalid input: Starting table should be less than or equal to ending table.\n");

return 1;

}

for (i = startTable; i <= endTable; i++) {

printf("Table of %d:\n", i);

for (j = 1; j <= 10; j++) {

printf("%d x %d = %d\n", i, j, i \* j);

}

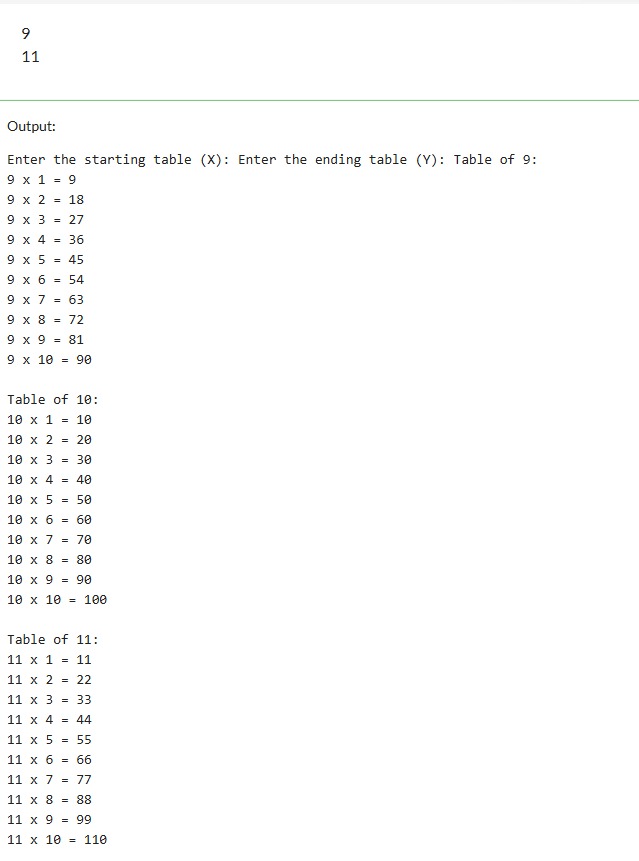
printf("\n");

}

return 0;

}

**Output: -**



**7) Write a program to compute factorial of a number using iterative approach.**

**Sol: -**

#include <stdio.h>

long long factorial(int n) {

long long fact = 1;

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

fact \*= i;

}

return fact;

}

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

if (num < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

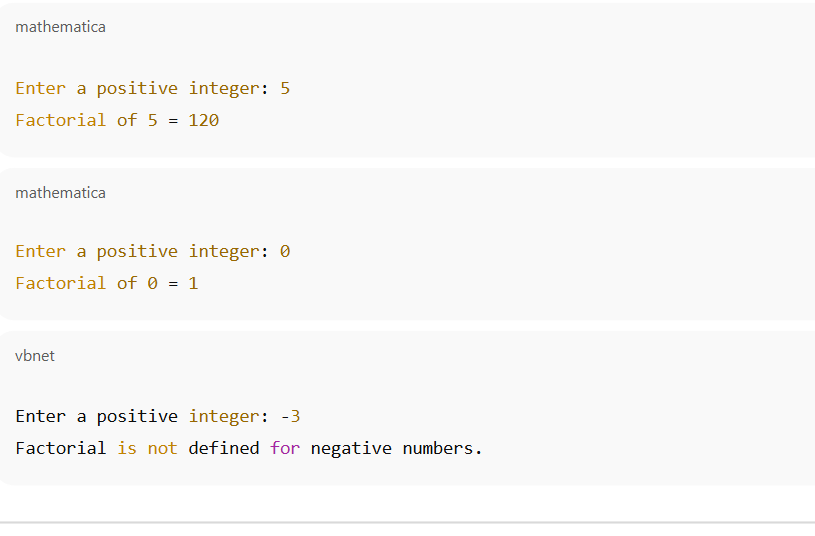
printf("Factorial of %d = %lld\n", num, factorial(num));

}

return 0;

}

**Output: -**

****

**8) Write a program to swap two numbers using functions.**

**Sol: -**

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int x, y;

printf("Enter two numbers: ");

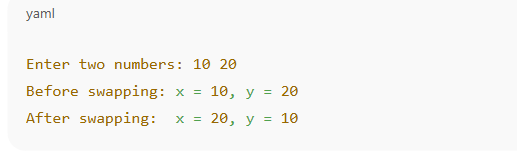
scanf("%d %d", &x, &y);

printf("Before swapping: x = %d, y = %d\n", x, y);

swap(&x, &y);

printf("After swapping: x = %d, y = %d\n", x, y)

return 0;}

****

**9) Write a function that returns the first integer between n\_min and n\_max entered as data to the calling function (main).**

**Sol: -**

#include <stdio.h>

int firstBetween(int n\_min, int n\_max) {

if (n\_max - n\_min > 1) {

return n\_min + 1;

} else {

return -1;

}

}

int main() {

int n\_min, n\_max, result;

printf("Enter n\_min and n\_max: ");

scanf("%d %d", &n\_min, &n\_max);

result = firstBetween(n\_min, n\_max);

if (result == -1) {

printf("No integer exists between %d and %d\n", n\_min, n\_max);

} else {

printf("The first integer between %d and %d is %d\n", n\_min, n\_max, result);

}

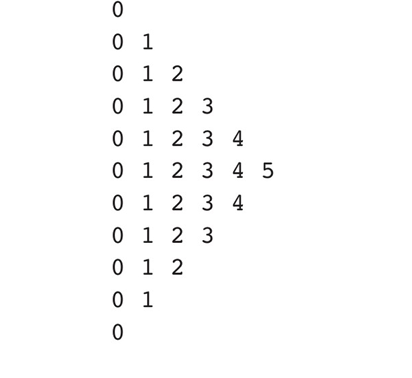
return 0;

}

**Output: -**

****

**10)Write nests of loops that cause the following output to be displayed.**

****

**Sol: -**

#include <stdio.h>

int main() {

int i, j;

for (i = 1; i <= 5; i++) {

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

printf("\*");

}

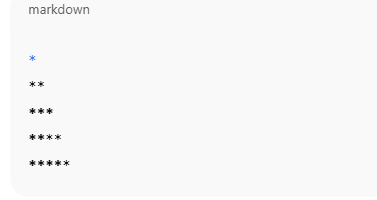
printf("\n");

}

return 0;

}

**Output: -**

****

#include <stdio.h>

int main() {

int i, j;

for (i = 5; i >= 1; i--) {

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

printf("\*");

}

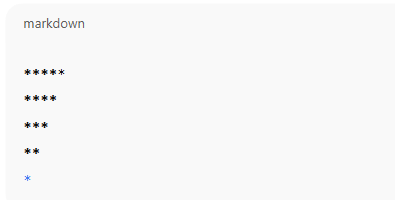
printf("\n");

}

return 0;

}

**Output: -**

****

**ASSIGNMENT-2**

1. **Write a program to check whether a given number is present in an array or not (Linear search).**

**Sol: -**

#include <stdio.h>

int main() {

int n, i, key, found = 0;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

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

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

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

}

printf("Enter the number to search: ");

scanf("%d", &key);

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

if (arr[i] == key) {

found = 1;

break;

}

}

if (found == 1)

printf("Number %d is present at position %d (index %d).\n", key, i + 1, i);

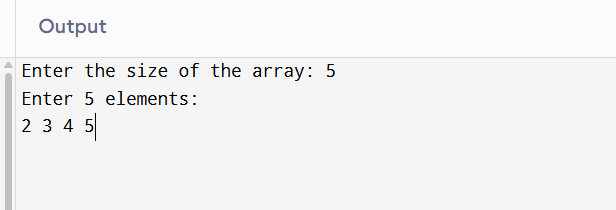
else

printf("Number %d is not present in the array.\n", key);

return 0;

}

**Output: -**

****

1. **Write a program to get second maximum and second minimum elements in an array.**

**Sol: -**

#include <stdio.h>

int main() {

int n, i, j, temp;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

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

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;

}

}

}

int secondMin = arr[1];

int secondMax = arr[n - 2];

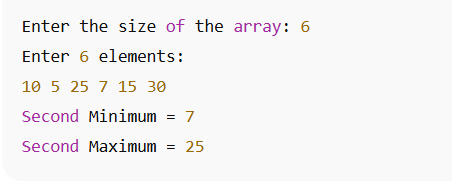
printf("Second Minimum = %d\n", secondMin);

printf("Second Maximum = %d\n", secondMax);

return 0;

}

**Output: -**

****

1. **Write a program to perform insertion (any location), deletion (any location) and traversal in an array.**

**Sol: -**

#include <stdio.h>

int main() {

int arr[100], n, i, pos, val, choice;

printf("Enter the number of elements in the array: ");

scanf("%d", &n);

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

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

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

}

while (1) {

printf("\n--- Menu ---\n");

printf("1. Traverse (Display Array)\n");

printf("2. Insert Element at any position\n");

printf("3. Delete Element from any position\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Array elements: ");

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

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

}

printf("\n");

break;

case 2:

printf("Enter position to insert (1 to %d): ", n + 1);

scanf("%d", &pos);

printf("Enter value to insert: ");

scanf("%d", &val);

if (pos < 1 || pos > n + 1) {

printf("Invalid position!\n");

} else {

for (i = n; i >= pos; i--) {

arr[i] = arr[i - 1];

}

arr[pos - 1] = val;

n++;

printf("Element inserted successfully.\n");

}

break;

case 3: // Deletion

printf("Enter position to delete (1 to %d): ", n);

scanf("%d", &pos);

if (pos < 1 || pos > n) {

printf("Invalid position!\n");

} else {

for (i = pos - 1; i < n - 1; i++) {

arr[i] = arr[i + 1];

}

n--;

printf("Element deleted successfully.\n");

}

break;

case 4:

return 0;

default:

printf("Invalid choice! Try again.\n");

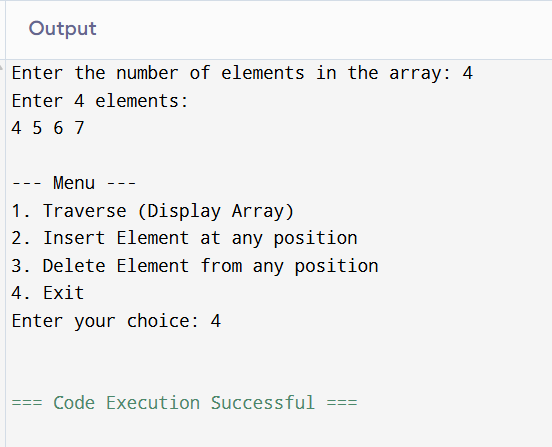
}

}

return 0;

}

**Output: -**

****

1. **Write a menu driven program to perform addition, multiplication and subtraction of 2 arrays.**

**Sol: -**

#include <stdio.h>

int main() {

int n, i, choice;

printf("Enter the size of the arrays: ");

scanf("%d", &n);

int arr1[n], arr2[n], result[n];

printf("Enter %d elements of first array:\n", n);

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

scanf("%d", &arr1[i]);

}

printf("Enter %d elements of second array:\n", n);

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

scanf("%d", &arr2[i]);

}

while (1) {

printf("\n--- Menu ---\n");

printf("1. Addition\n");

printf("2. Subtraction\n");

printf("3. Multiplication\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Result (Addition): ");

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

result[i] = arr1[i] + arr2[i];

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

}

printf("\n");

break;

case 2:

printf("Result (Subtraction): ");

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

result[i] = arr1[i] - arr2[i];

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

}

printf("\n");

break;

case 3:

printf("Result (Multiplication): ");

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

result[i] = arr1[i] \* arr2[i];

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

}

printf("\n");

break;

case 4:

return 0;

default:

printf("Invalid choice! Try again.\n");

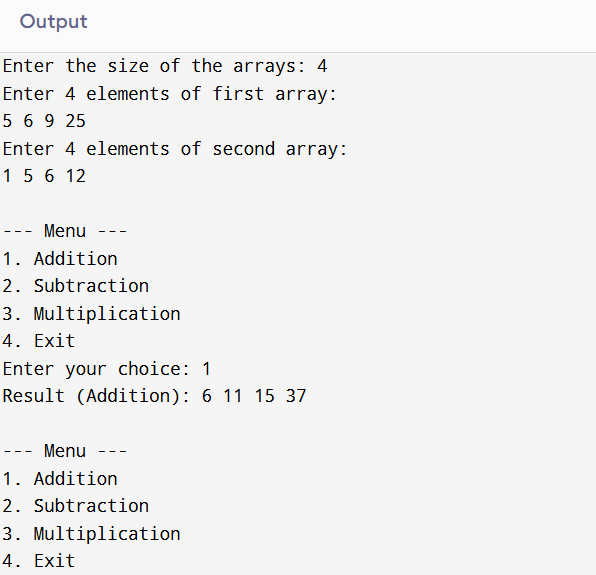
}

}

return 0;

}

**Output: -**

****

1. **Write a program to perform sorting while merging (Merge two sorted arrays into one sorted array).**

**Sol: -**

#include <stdio.h>

int main() {

int n1, n2, n3, i, j, k;

printf("Enter size of first sorted array: ");

scanf("%d", &n1);

printf("Enter size of second sorted array: ");

scanf("%d", &n2);

int arr1[n1], arr2[n2], merged[n1 + n2];

printf("Enter %d elements (sorted) for first array:\n", n1);

for (i = 0; i < n1; i++) {

scanf("%d", &arr1[i]);

}

printf("Enter %d elements (sorted) for second array:\n", n2);

for (i = 0; i < n2; i++) {

scanf("%d", &arr2[i]);

}

i = j = k = 0;

while (i < n1 && j < n2) {

if (arr1[i] <= arr2[j]) {

merged[k++] = arr1[i++];

} else {

merged[k++] = arr2[j++];

}

}

while (i < n1) {

merged[k++] = arr1[i++];

}

while (j < n2) {

merged[k++] = arr2[j++];

}

n3 = n1 + n2;

printf("Merged Sorted Array: ");

for (i = 0; i < n3; i++) {

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

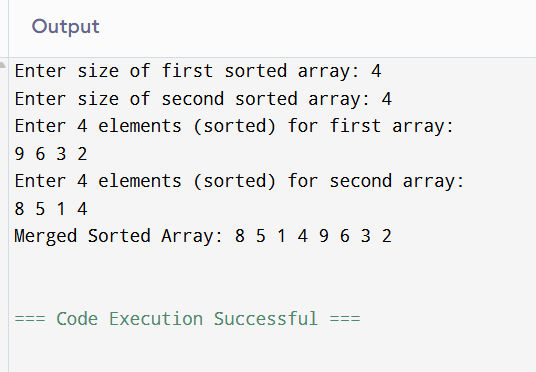
}

printf("\n");

return 0;

}

**Output: -**

****

1. **Write the above programs (1,2, and 3) using functions and call by address only.**

**Sol:-**

#include <stdio.h>

int linearSearch(int \*arr, int n, int key) {

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

if (\*(arr + i) == key) {

return i;

}

}

return -1;

}

int main() {

int n, key, pos;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

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

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

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

}

printf("Enter the number to search: ");

scanf("%d", &key);

pos = linearSearch(arr, n, key);

if (pos != -1)

printf("Number %d found at position %d (index %d).\n", key, pos + 1, pos);

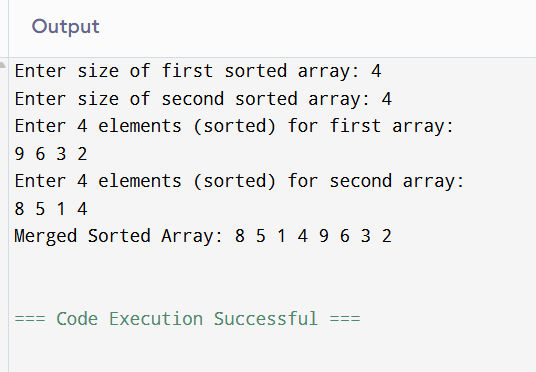
else

printf("Number %d not found in the array.\n", key);

return 0;

}

**Output: -**

****

**ASSIGNMENT-3**

1. **Write a program to implement strlen() function.**

**Sol: -**

#include <stdio.h>

int my\_strlen(const char \*str) {

int length = 0;

while (str[length] != '\0') {

length++;

}

return length;

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

if (str[my\_strlen(str) - 1] == '\n') {

str[my\_strlen(str) - 1] = '\0';

}

printf("Length of the string: %d\n", my\_strlen(str));

return 0;

}

**Output: -**



1. **Write a program to implement strcpy() function.**

**Sol: -**

#include <stdio.h>

void my\_strcpy(char \*dest, const char \*src) {

int i = 0;

while (src[i] != '\0') {

dest[i] = src[i];

i++;

}

dest[i] = '\0';

}

int main() {

char source[100], destination[100];

printf("Enter a string: ");

fgets(source, sizeof(source), stdin);

int len = 0;

while (source[len] != '\0') {

if (source[len] == '\n') {

source[len] = '\0';

break;

}

len++;

}

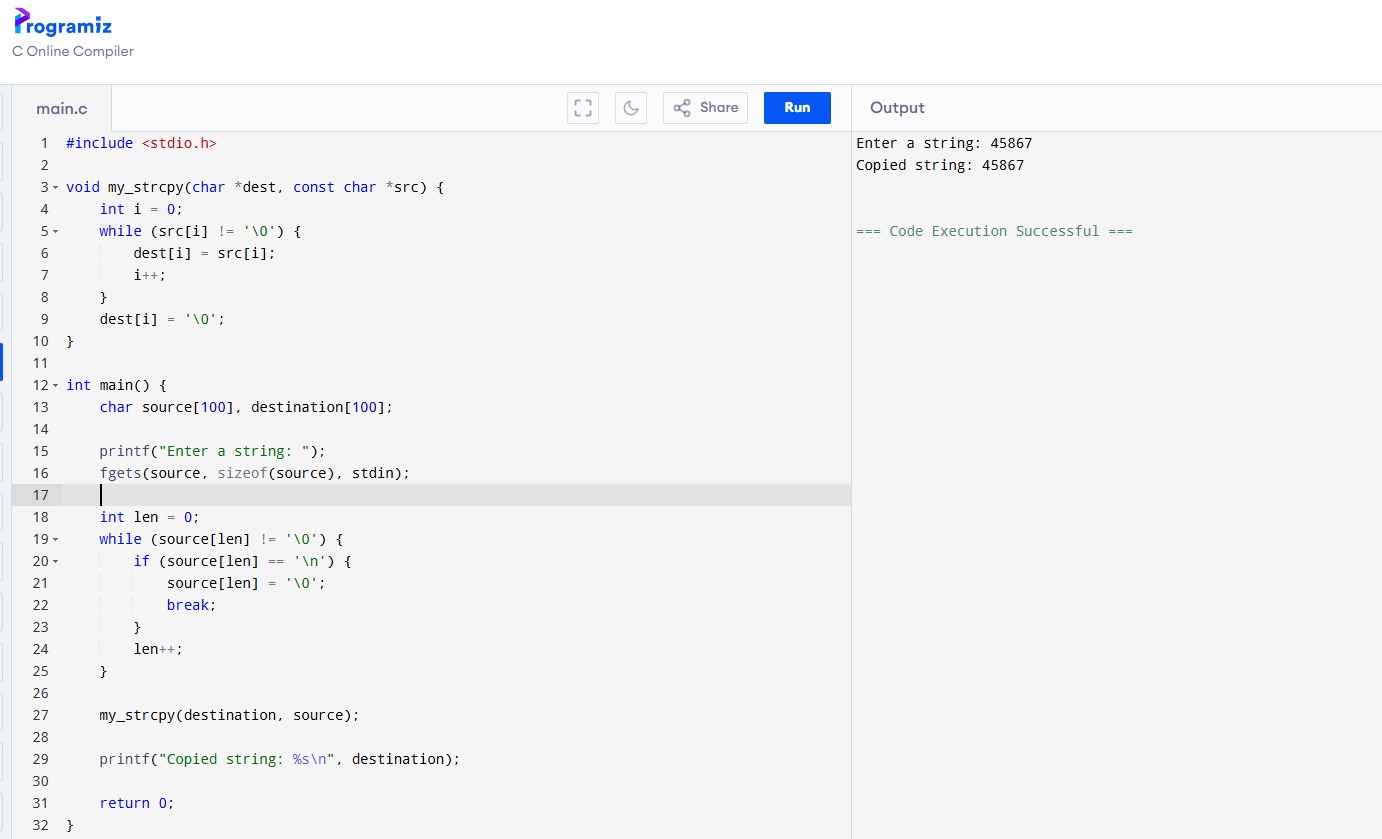
my\_strcpy(destination, source);

printf("Copied string: %s\n", destination);

return 0;

}

**Output: -**



1. **Write a program to implement strcat() function.**

**Sol: -**

#include <stdio.h>

void my\_strcpy(char \*dest, const char \*src) {

int i = 0;

while (src[i] != '\0') {

dest[i] = src[i];

i++;

}

dest[i] = '\0';

}

int main() {

char source[100], destination[100];

printf("Enter a string: ");

fgets(source, sizeof(source), stdin);

int len = 0;

while (source[len] != '\0') {

if (source[len] == '\n') {

source[len] = '\0';

break;

}

len++;

}

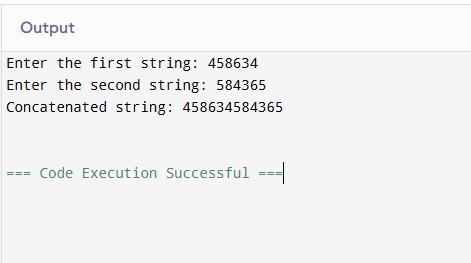
my\_strcpy(destination, source);

printf("Copied string: %s\n", destination);

return 0;

}

**Output: -**



1. **Write a program to implement strcmp() function.**

**Sol: -**

#include <stdio.h>

int my\_strcmp(const char \*str1, const char \*str2) {

int i = 0;

while (str1[i] != '\0' && str2[i] != '\0') {

if (str1[i] != str2[i]) {

return (unsigned char)str1[i] - (unsigned char)str2[i];

}

i++;

}

return (unsigned char)str1[i] - (unsigned char)str2[i];

}

int main() {

char str1[100], str2[100];

printf("Enter first string: ");

fgets(str1, sizeof(str1), stdin);

printf("Enter second string: ");

fgets(str2, sizeof(str2), stdin);

int len = 0;

while (str1[len] != '\0') {

if (str1[len] == '\n') {

str1[len] = '\0';

break;

}

len++;

}

len = 0;

while (str2[len] != '\0') {

if (str2[len] == '\n') {

str2[len] = '\0';

break;

}

len++;

}

int result = my\_strcmp(str1, str2);

if (result == 0) {

printf("Strings are equal.\n");

} else if (result < 0) {

printf("First string is less than second string.\n");

} else {

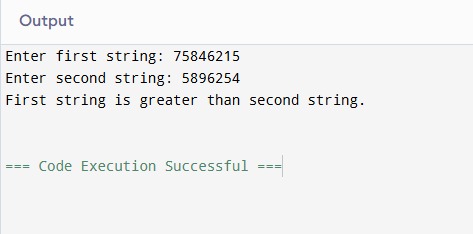
printf("First string is greater than second string.\n");

}

return 0;

}

**Output: -**



1. **WAP to demonstrate limitations of Two-Dimensional Array of Characters.**

**Sol: -**

#include <stdio.h>

#include <string.h>

int main() {

char arr[3][6];

printf("Enter 3 strings (each max 5 characters):\n");

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

fgets(arr[i], sizeof(arr[i]), stdin);

size\_t len = strlen(arr[i]);

if (len > 0 && arr[i][len - 1] == '\n') {

arr[i][len - 1] = '\0';

}

}

printf("\nStrings stored in 2D array:\n");

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

printf("String %d: %s\n", i + 1, arr[i]);

}

printf("\nTry to input a longer string (more than 5 chars):\n");

printf("Enter a string: ");

char temp[50];

fgets(temp, sizeof(temp), stdin);

strncpy(arr[0], temp, 5);

arr[0][5] = '\0'; // manually terminate

printf("After copying longer string forcibly into arr[0]: %s\n", arr[0]);

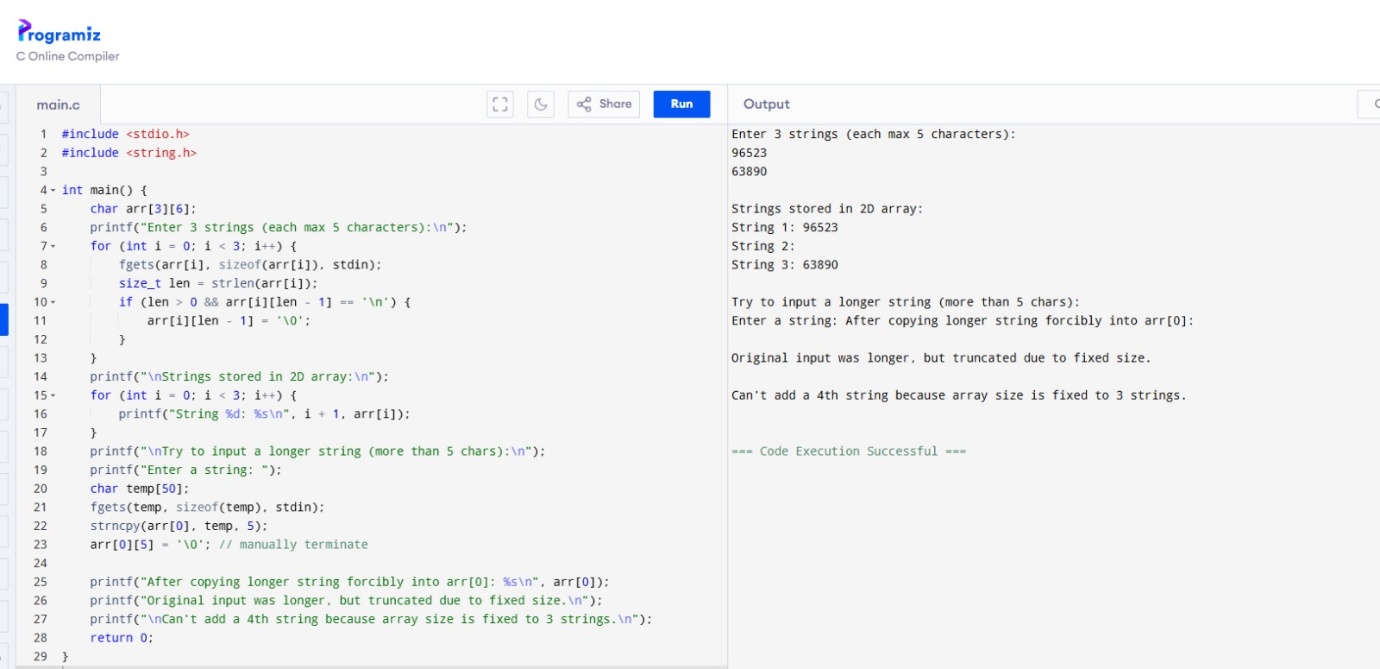
printf("Original input was longer, but truncated due to fixed size.\n");

printf("\nCan't add a 4th string because array size is fixed to 3 strings.\n");

return 0;

}

**Output: -**



1. **WAP to demonstrate an array of Pointers to Strings.**

**Sol: -**

#include <stdio.h>

int main() {

const char \*arr[] = {

"Hello",

"World",

"Pointers",

"to",

"Strings"

};

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

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

printf("%s\n", arr[i]);

}

return 0;

}

**Output: -**

