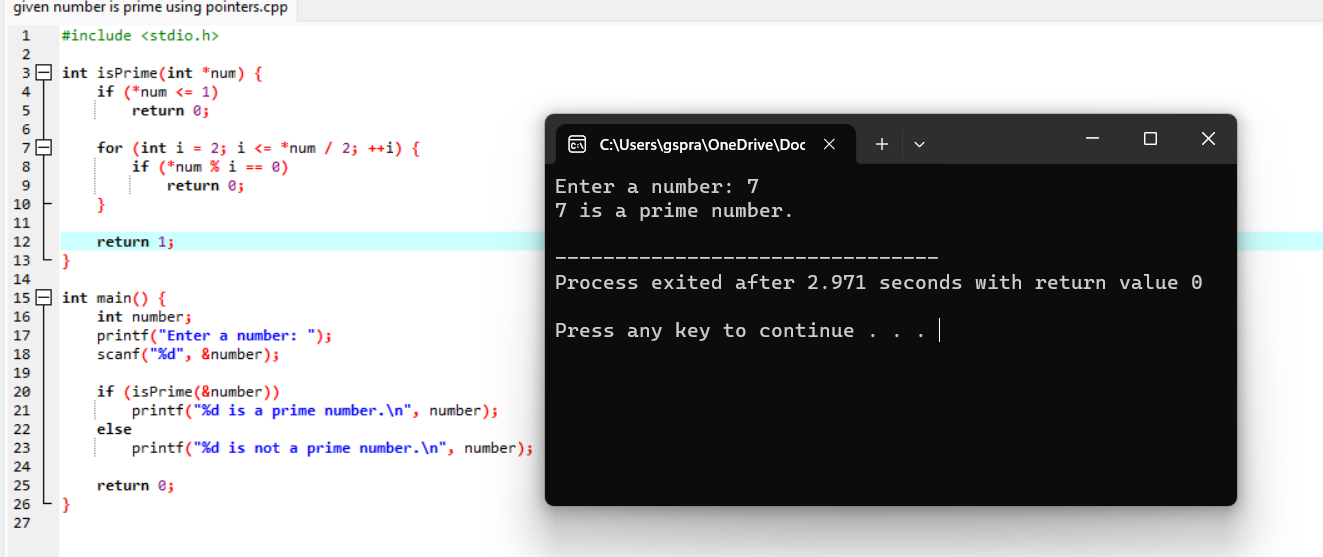
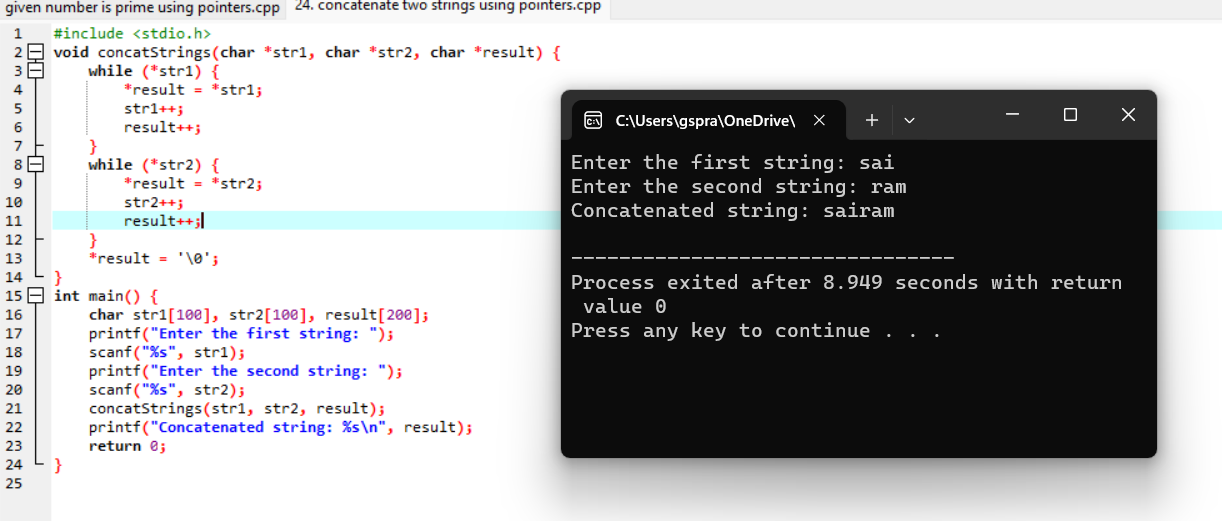
**24/02/2024**

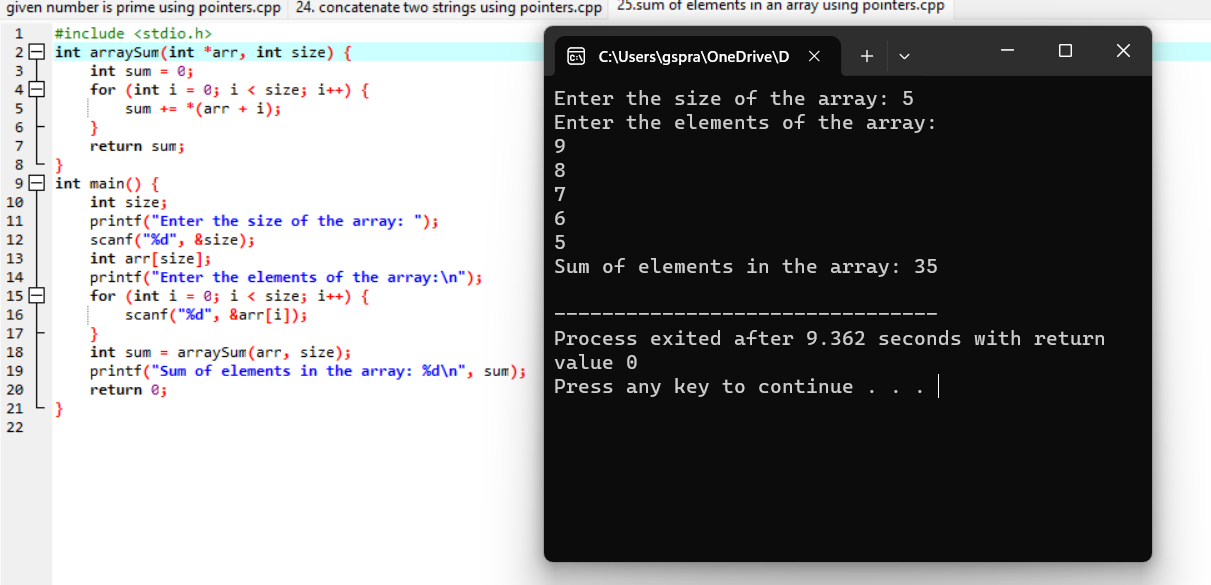
**23. Given number is prime using pointer:**

****

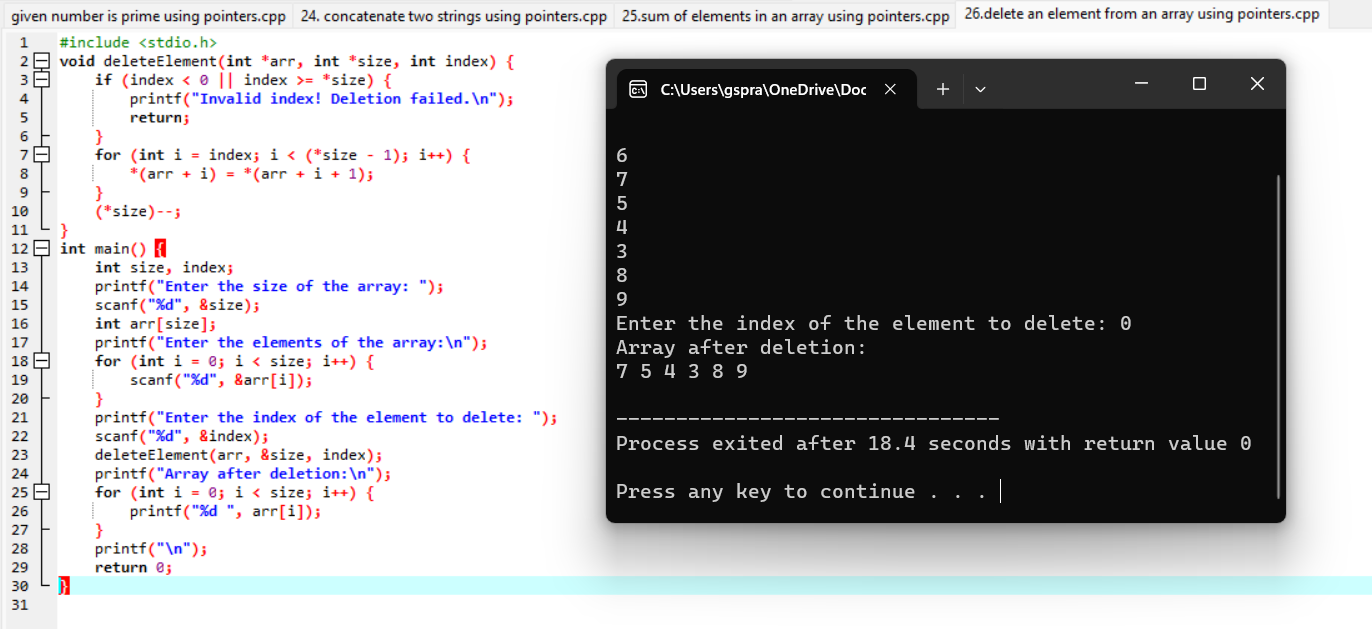
**24.** **concatenate two strings using pointers:**

****

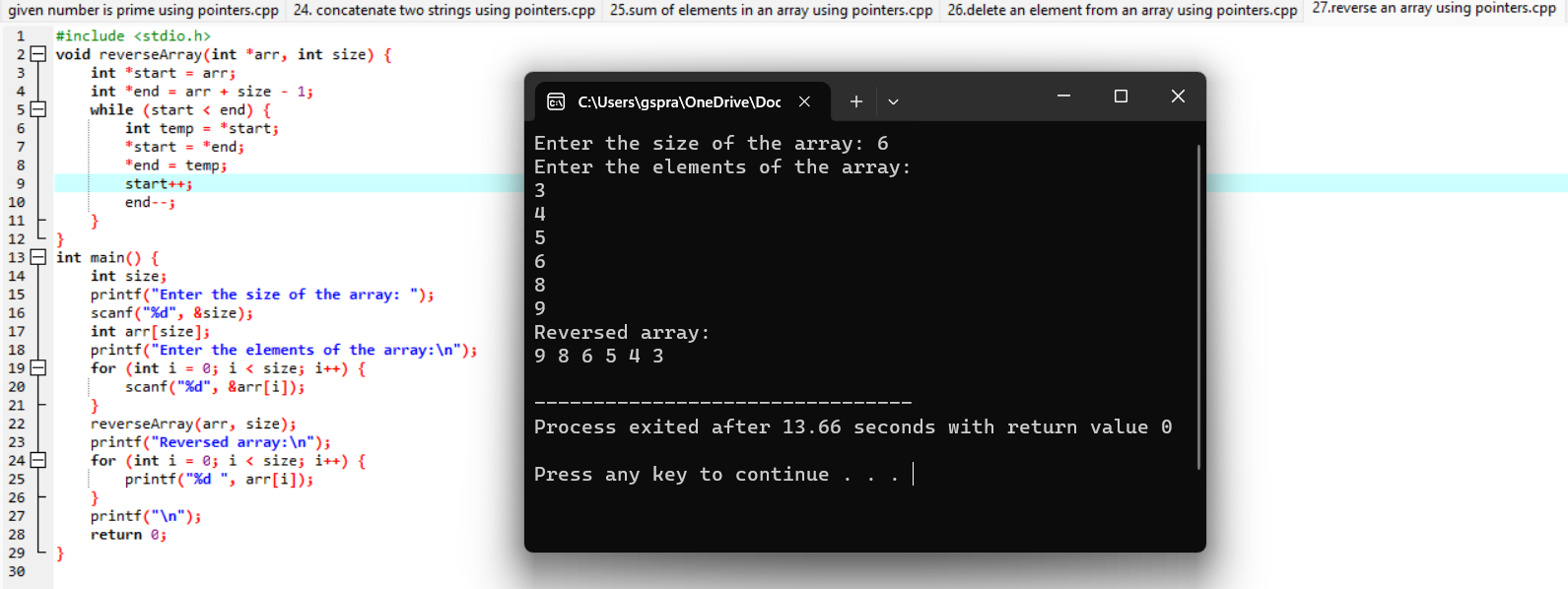
**25.** **sum of elements in an array using pointers:**

****

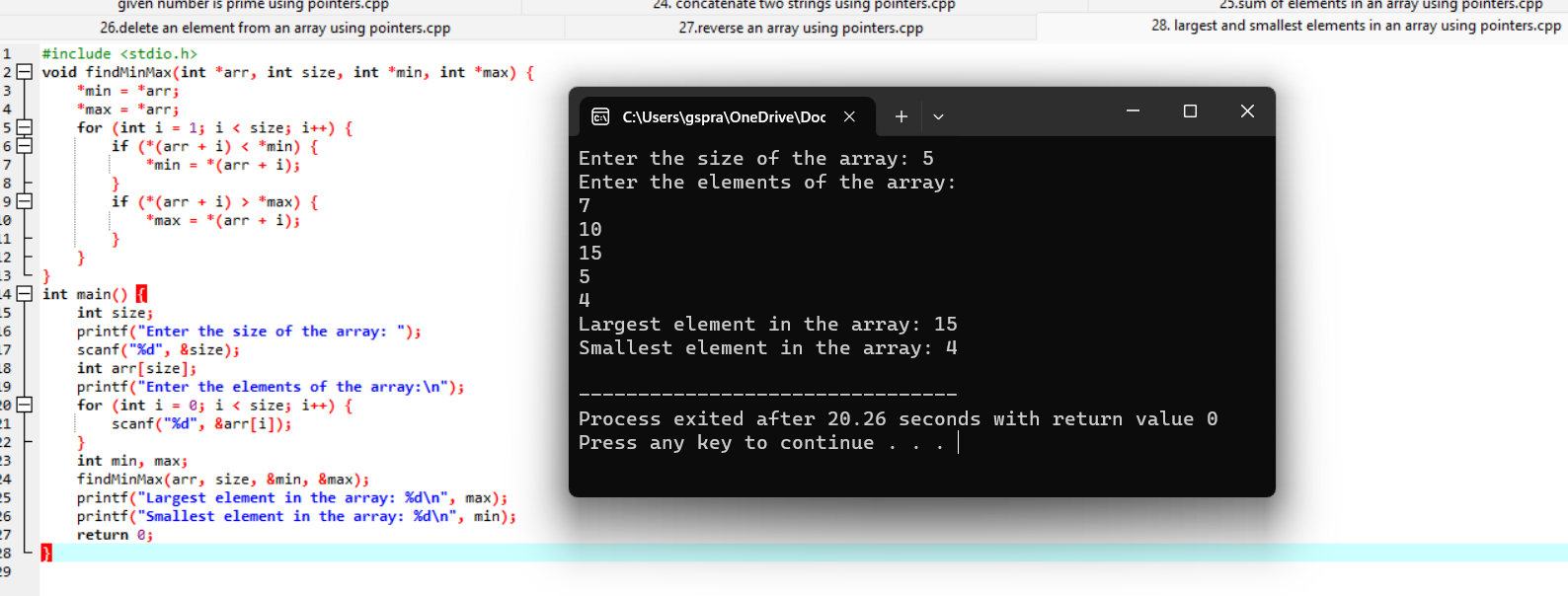
**26.** **delete an element from an array using pointers:**

****

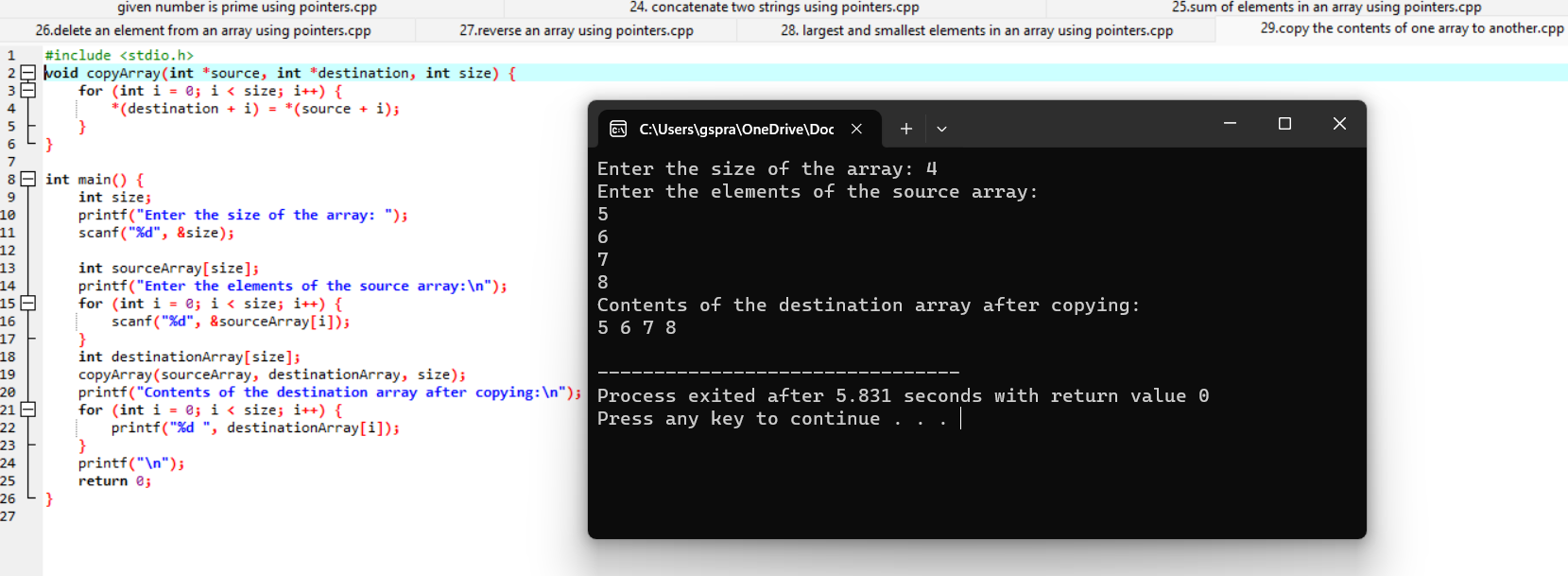
**27.** **reverse an array using pointers:**

****

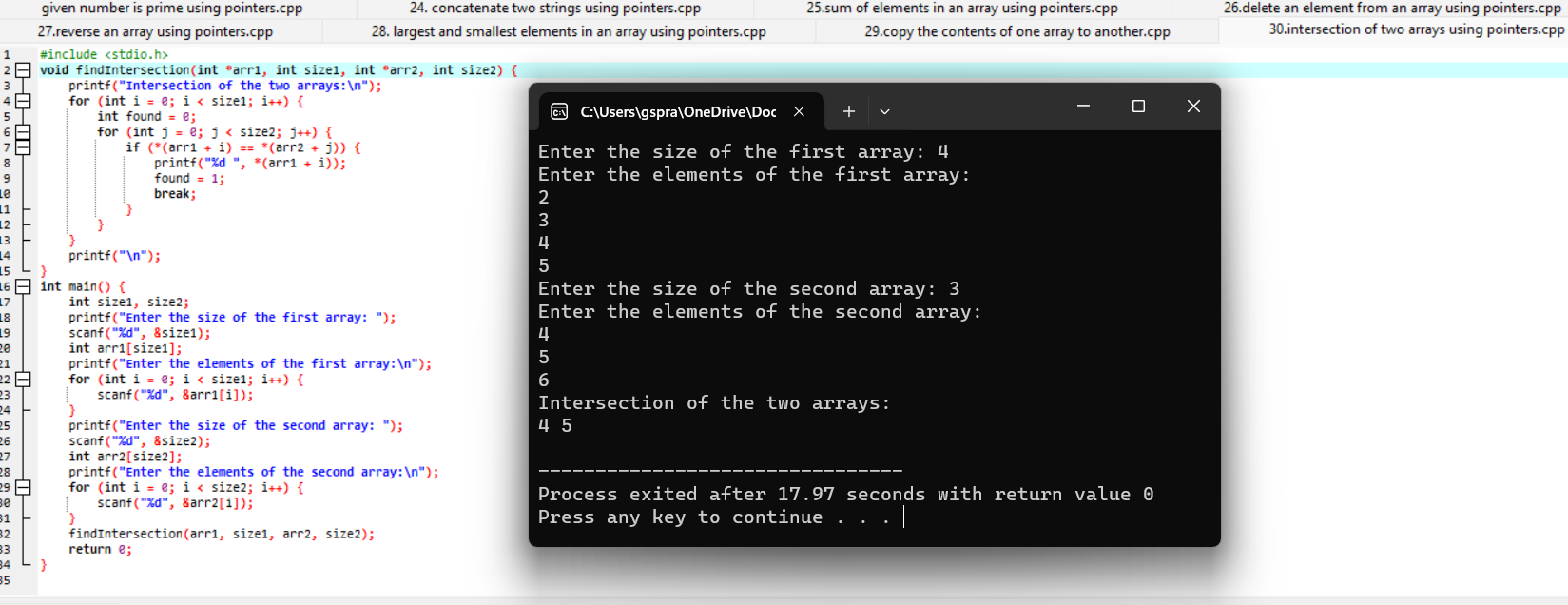
**28**. **largest and smallest elements in an array using pointers:**

****

**29.** **copy the contents of one array to another:**

****

**30.** **intersection of two arrays using pointers:**

****

**Analytical programs:**

**21. The area of a triangle given its three sides:**

#include <stdio.h>

#include <math.h>

double calculateArea(double a, double b, double c) {

double s = (a + b + c) / 2.0;

double area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

int main() {

double side1, side2, side3;

printf("Enter the lengths of the three sides of the triangle: ");

scanf("%lf %lf %lf", &side1, &side2, &side3);

if (side1 + side2 > side3 && side2 + side3 > side1 && side3 + side1 > side2) {

double area = calculateArea(side1, side2, side3);

printf("Area of the triangle: %.2lf\n", area);

} else {

printf("Invalid triangle sides. Triangle cannot be formed.\n");

}

return 0;

}

**Output:**

Enter the lengths of the three sides of the triangle:

7

5

6

Area of the triangle: 14.70

**22.Given year is leap or not a leap year:**

#include <stdio.h>

int isLeapYear(int year) {

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

return 1;

} else {

return 0;

}

}

int main() {

int year;

printf("Enter a year: ");

scanf("%d", &year);

if (isLeapYear(year)) {

printf("%d is a leap year.\n", year);

} else {

printf("%d is not a leap year.\n", year);

}

return 0;

}

**Output:**

Enter a year: 2024

2024 is a leap year.

**23. Temperature from Celsius to Fahrenheit and vice versa.**

#include <stdio.h>

double celsiusToFahrenheit(double celsius) {

return (celsius \* 9 / 5) + 32;

}

double fahrenheitToCelsius(double fahrenheit) {

return (fahrenheit - 32) \* 5 / 9;

}

int main() {

double temperature;

char choice;

printf("Enter the temperature: ");

scanf("%lf", &temperature);

printf("Enter 'C' to convert to Celsius or 'F' to convert to Fahrenheit: ");

scanf(" %c", &choice);

switch (choice) {

case 'C':

case 'c':

printf("%.2lf Fahrenheit is equal to %.2lf Celsius.\n", temperature, fahrenheitToCelsius(temperature));

break;

case 'F':

case 'f':

printf("%.2lf Celsius is equal to %.2lf Fahrenheit.\n", temperature, celsiusToFahrenheit(temperature));

break;

default:

printf("Invalid choice.\n");

}

return 0;

}

**Output:**

Enter the temperature: 30

Enter 'C' to convert to Celsius or 'F' to convert to Fahrenheit: f

30.00 Celsius is equal to 86.00 Fahrenheit**.**

**24.** **calculates the factorial of a given number using recursion:**

#include <stdio.h>

unsigned long long factorial(int n) {

if (n == 0) {

return 1;

}

return n \* factorial(n - 1);

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (number < 0) {

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

} else {

printf("Factorial of %d is %llu\n", number, factorial(number));

}

return 0;

}

**Output:**

Enter a number: 7

Factorial of 7 is 5040

**25.** **sum of the first n natural numbers.**

#include <stdio.h>

int sumOfNaturalNumbers(int n) {

if (n == 0) {

return 0;

}

return n + sumOfNaturalNumbers(n - 1);

}

int main() {

int n;

printf("Enter a positive integer n: ");

scanf("%d", &n);

if (n <= 0) {

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

} else {

printf("Sum of the first %d natural numbers is %d.\n", n, sumOfNaturalNumbers(n));

}

return 0;

}#include <stdio.h>

int sumOfNaturalNumbers(int n) {

if (n == 0) {

return 0;

}

return n + sumOfNaturalNumbers(n - 1);

}

int main() {

int n;

printf("Enter a positive integer n: ");

scanf("%d", &n);

if (n <= 0) {

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

} else {

printf("Sum of the first %d natural numbers is %d.\n", n, sumOfNaturalNumbers(n));

}

return 0;

}

**Output:**

Enter a positive integer n: 9

Sum of the first 9 natural numbers is 45**.**

**26.** **given number is Armstrong number or not.**

#include <stdio.h>

#include <math.h>

int countDigits(int number) {

int count = 0;

while (number != 0) {

number /= 10;

count++;

}

return count;

}

int isArmstrong(int number) {

int originalNumber = number;

int numDigits = countDigits(number);

int sum = 0;

while (number != 0) {

int digit = number % 10;

sum += pow(digit, numDigits);

number /= 10;

}

return (sum == originalNumber);

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (isArmstrong(number)) {

printf("%d is an Armstrong number.\n", number);

} else {

printf("%d is not an Armstrong number.\n", number);

}

return 0;

}

**Output:**

Enter a number: 143

143 is not an Armstrong number.

Enter a number: 153

153 is an Armstrong number**.**

**27.** **roots of a quadratic equation ax^2 + bx + c = 0.**

#include <stdio.h>

#include <math.h>

void findRoots(double a, double b, double c) {

double discriminant, realPart, imaginaryPart;

double root1, root2;

discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0) {

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

printf("Roots are real and different.\n");

printf("Root 1 = %.2lf\n", root1);

printf("Root 2 = %.2lf\n", root2);

}

else if (discriminant == 0) {

root1 = root2 = -b / (2 \* a);

printf("Roots are real and equal.\n");

printf("Root 1 = Root 2 = %.2lf\n", root1);

}

else {

realPart = -b / (2 \* a);

imaginaryPart = sqrt(-discriminant) / (2 \* a);

printf("Roots are complex and different.\n");

printf("Root 1 = %.2lf + %.2lfi\n", realPart, imaginaryPart);

printf("Root 2 = %.2lf - %.2lfi\n", realPart, imaginaryPart);

}

}

int main() {

double a, b, c;

printf("Enter coefficients a, b, and c of the quadratic equation (ax^2 + bx + c = 0):\n");

scanf("%lf %lf %lf", &a, &b, &c);

findRoots(a, b, c);

return 0;

}

**Output:**

Enter coefficients a, b, and c of the quadratic equation (ax^2 + bx + c = 0):

1

-3

2

Roots are real and different.

Root 1 = 2.00

Root 2 = 1.00

**28.** **given string is a valid email address or not.**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

bool isValidLocalChar(char ch) {

return (ch >= 'a' && ch <= 'z') ||

(ch >= 'A' && ch <= 'Z') ||

(ch >= '0' && ch <= '9') ||

(ch == '!' || ch == '#' || ch == '$' || ch == '%' || ch == '&' ||

ch == '\'' || ch == '\*' || ch == '+' || ch == '-' || ch == '/' ||

ch == '=' || ch == '?' || ch == '^' || ch == '\_' || ch == '`' ||

ch == '{' || ch == '|' || ch == '}' || ch == '~' || ch == '.');

}

bool isValidDomainChar(char ch) {

return (ch >= 'a' && ch <= 'z') ||

(ch >= 'A' && ch <= 'Z') ||

(ch >= '0' && ch <= '9') ||

(ch == '-' || ch == '.');

}

bool isValidEmail(char \*email) {

int length = strlen(email);

int atIndex = -1;

bool dotAfterAt = false;

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

if (email[i] == '@') {

if (i == 0 || i == length - 1) {

return false;

}

atIndex = i;

break;

}

}

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

if (!isValidLocalChar(email[i])) {

return false;

}

}

for (int i = atIndex + 1; i < length; i++) {

if (email[i] == '.') {

if (i == atIndex + 1 || i == length - 1) {

return false;

}

dotAfterAt = true;

} else {

if (!isValidDomainChar(email[i])) {

return false;

}

}

}

return dotAfterAt;

}

int main() {

char email[100];

printf("Enter an email address: ");

scanf("%s", email);

if (isValidEmail(email)) {

printf("%s is a valid email address.\n", email);

} else {

printf("%s is not a valid email address.\n", email);

}

return 0;

}

**Output:**

Enter an email address: sv77@gmail.com

sv77@gmail.com is a valid email address.

**29. sum of digits of a given number until it becomes a single-digit number.**

#include <stdio.h>

int digitalRoot(int num) {

if (num < 10) {

return num;

}

int sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

return digitalRoot(sum);

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

int root = digitalRoot(number);

printf("The digital root of %d is %d.\n", number, root);

return 0;

}

**Output:**

Enter a number: 9121

The digital root of 9121 is 4.

**30. number is a palindrome or not without converting it to a string.**

#include <stdio.h>

int isPalindrome(int num) {

int reversedNum = 0;

int originalNum = num;

while (num > 0) {

int digit = num % 10;

reversedNum = reversedNum \* 10 + digit;

num /= 10;

}

return (reversedNum == originalNum);

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (isPalindrome(number)) {

printf("%d is a palindrome.\n", number);

} else {

printf("%d is not a palindrome.\n", number);

}

return 0;

}

**Output:**

Enter a number: 143

143 is not a palindrome.