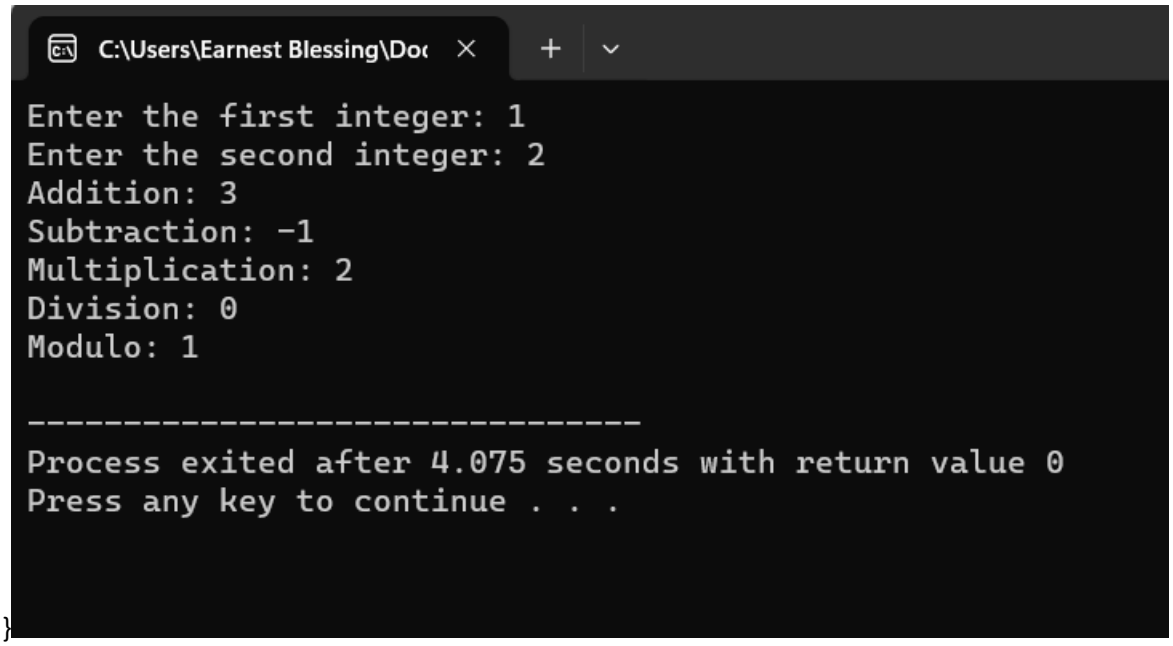
1. Write a program to read in two integers and perform the following operations on them: addition, subtraction, multiplication, division, and modulo.

```
#include <iostream>


int main() {

    // Declare variables to store input values

    int firstNumber, secondNumber;


    // Read in two integers

    std::cout << "Enter the first integer: ";

    std::cin >> firstNumber;


    std::cout << "Enter the second integer: ";

    std::cin >> secondNumber;


    // Perform operations

    int additionResult = firstNumber + secondNumber;

    int subtractionResult = firstNumber - secondNumber;


    // Check for division by zero before performing division

    if (secondNumber != 0) {

        int multiplicationResult = firstNumber * secondNumber;

        int divisionResult = firstNumber / secondNumber;

        int moduloResult = firstNumber % secondNumber;


        // Display results

        std::cout << "Addition: " << additionResult << std::endl;

        std::cout << "Subtraction: " << subtractionResult << std::endl;

        std::cout << "Multiplication: " << multiplicationResult << std::endl;

        std::cout << "Division: " << divisionResult << std::endl;

        std::cout << "Modulo: " << moduloResult << std::endl;
```

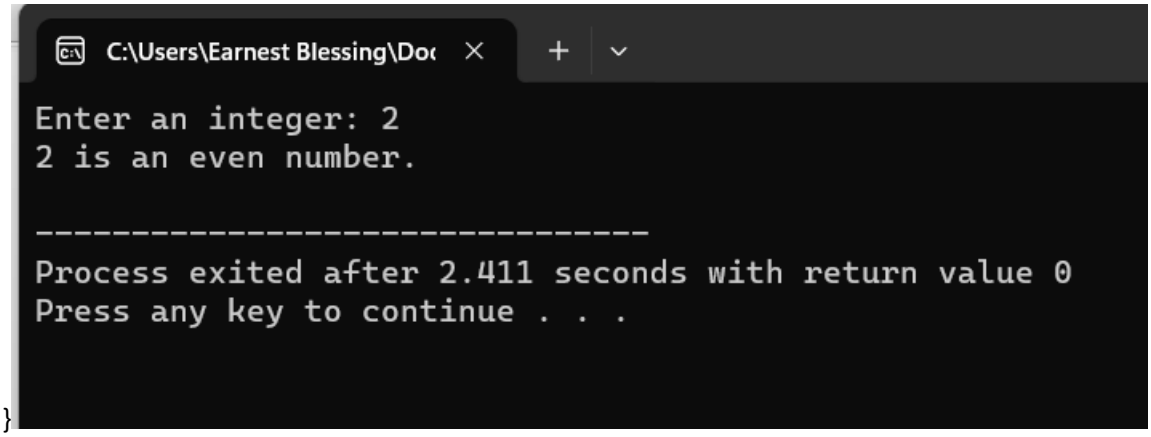
```
    } else {
        std::cout << "Error: Division by zero is not allowed." << std::endl;
    }

    return 0;
```




```
}
```

2. Program to determine the integer is odd or even

```
#include <iostream>

int main() {
    int number;
    std::cout << "Enter an integer: ";
    std::cin >> number;
    if (number % 2 == 0) {
        std::cout << number << " is an even number." << std::endl;
    } else {
        std::cout << number << " is an odd number." << std::endl;
    }
    return 0;
```

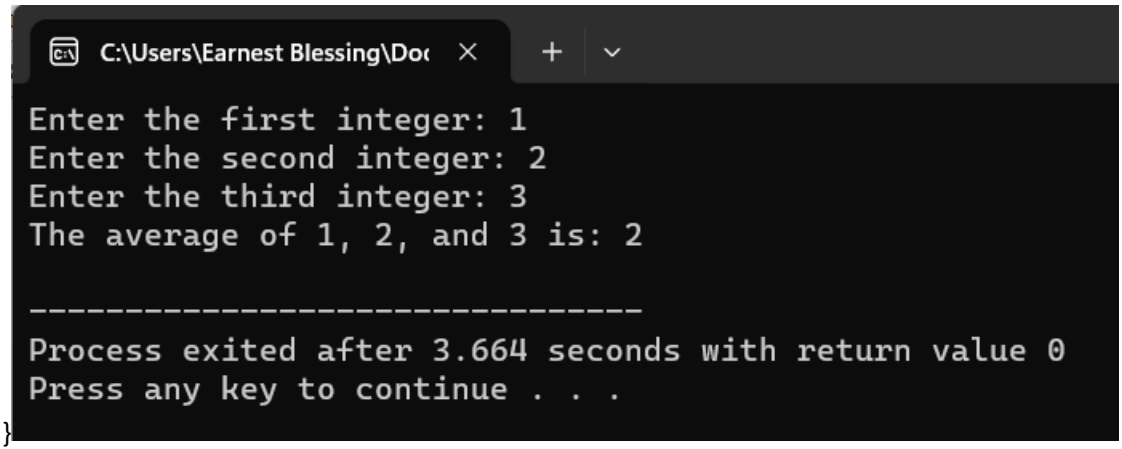


}

3. Program to compute the average of three integers

```
#include <iostream>


int main() {

   // Declare variables to store user input

   int num1, num2, num3;


   // Read three integers from the user

   std::cout << "Enter the first integer: ";

   std::cin >> num1;


   std::cout << "Enter the second integer: ";

   std::cin >> num2;


   std::cout << "Enter the third integer: ";

   std::cin >> num3;


   // Calculate the average

   double average = static_cast<double>(num1 + num2 + num3) / 3;


   // Display the result

   std::cout << "The average of " << num1 << ", " << num2 << ", and " << num3 << " is: " << average <<
std::endl;
```

```
    return 0;
```

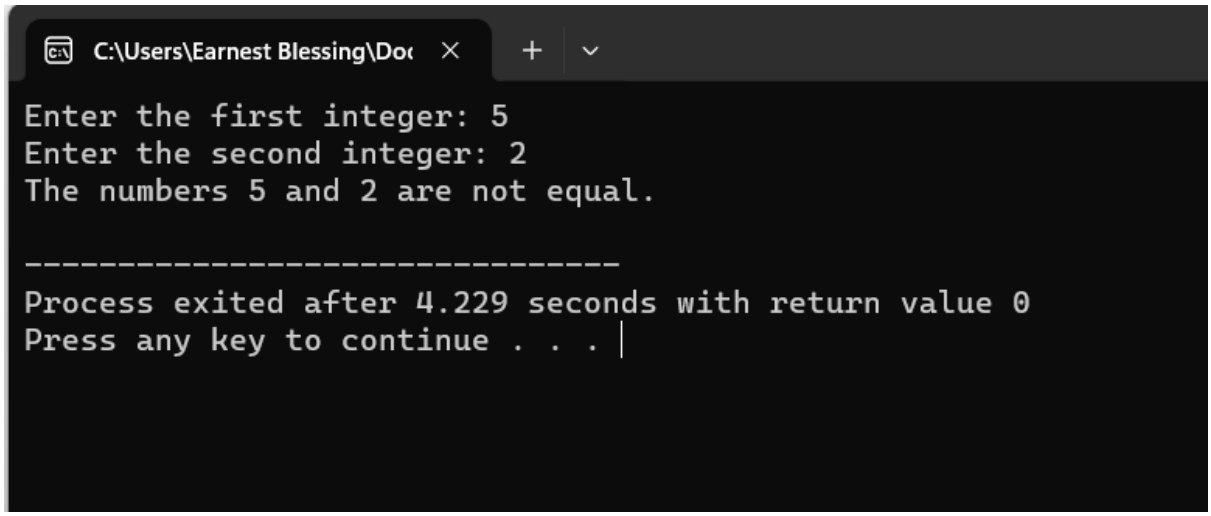



```
}
```

4. Program to check two numbers are equal or not

```
#include <iostream>

int main() {
    // Declare variables to store user input
    int num1, num2;

    // Read two integers from the user
    std::cout << "Enter the first integer: ";
    std::cin >> num1;

    std::cout << "Enter the second integer: ";
    std::cin >> num2;

    // Check if the numbers are equal
    if (num1 == num2) {
        std::cout << "The numbers " << num1 << " and " << num2 << " are equal." << std::endl;
    } else {
        std::cout << "The numbers " << num1 << " and " << num2 << " are not equal." << std::endl;
    }
```

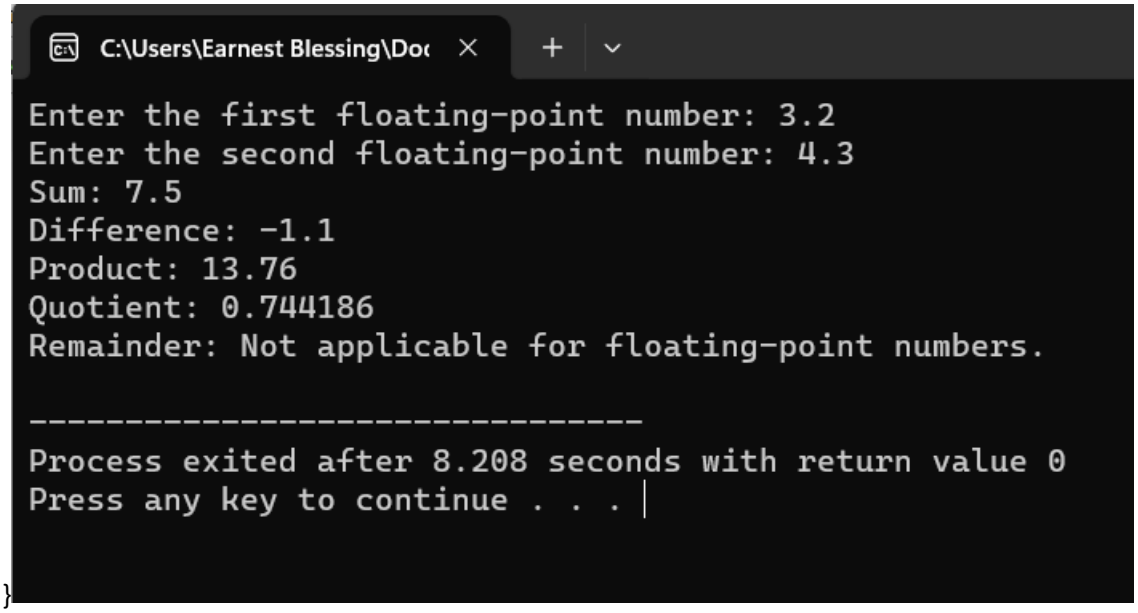
```
    return 0;
}
```



5. Write a program to read in two Floating numbers and perform the following operations on them: addition, subtraction, multiplication, division, and modulo.

```
#include <iostream>


int main() {
    // Declare variables to store user input
    double num1, num2;

    // Read two floating-point numbers from the user
    std::cout << "Enter the first floating-point number: ";
    std::cin >> num1;

    std::cout << "Enter the second floating-point number: ";
    std::cin >> num2;

    // Perform operations and display results
    std::cout << "Sum: " << num1 + num2 << std::endl;
    std::cout << "Difference: " << num1 - num2 << std::endl;
    std::cout << "Product: " << num1 * num2 << std::endl;
```

```
    // Check if the second number is not zero before performing division and modulo
    if (num2 != 0) {
        std::cout << "Quotient: " << num1 / num2 << std::endl;
        std::cout << "Remainder: Not applicable for floating-point numbers." << std::endl;
    } else {
        std::cout << "Cannot perform division because the second number is zero." << std::endl;
    }

    return 0;
```

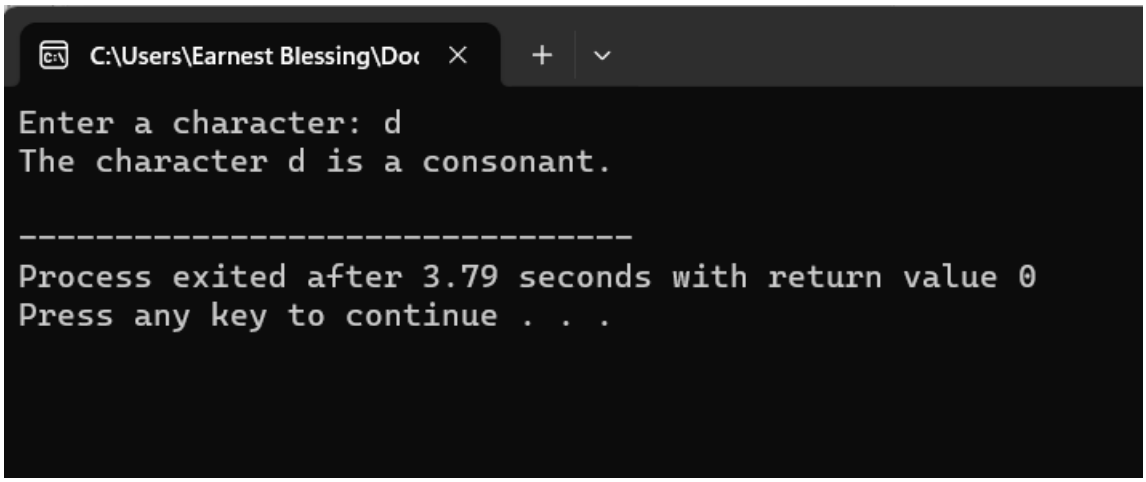


}

6. Program to check the character is a vowel or consonant

```
#include <iostream>

int main() {
    // Declare a variable to store user input
    char ch;

    // Read a character from the user
    std::cout << "Enter a character: ";
```

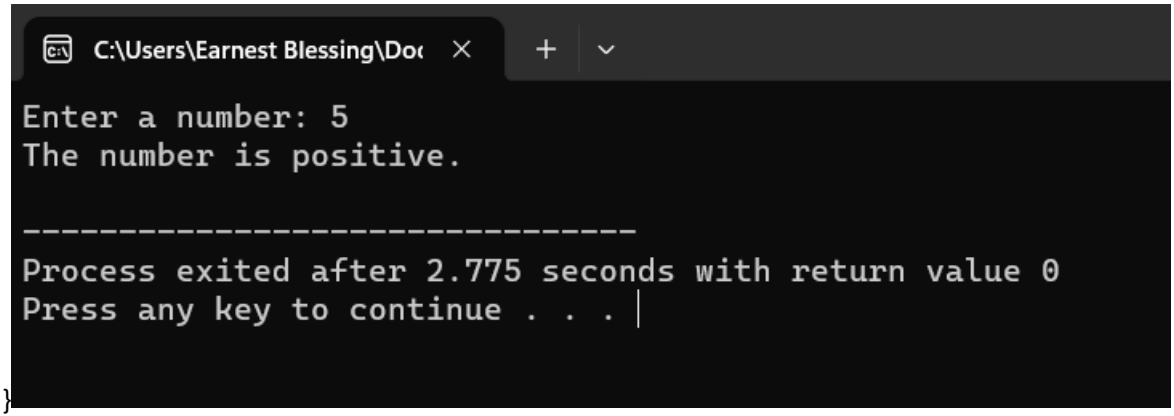
```
    std::cin >> ch;

    // Check if the character is a vowel
    if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
        ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
        std::cout << "The character " << ch << " is a vowel." << std::endl;
    } else {
        std::cout << "The character " << ch << " is a consonant." << std::endl;
    }

    return 0;
}
```



7. Program to check the number is positive, negative or zero

```
#include <iostream>

int main() {
    // Declare variable
    double number;

    // Read a number from the user
    std::cout << "Enter a number: ";
    std::cin >> number;
```

```
    // Check whether the number is positive, negative, or zero
    if (number > 0) {
        std::cout << "The number is positive." << std::endl;
    } else if (number < 0) {
        std::cout << "The number is negative." << std::endl;
    } else {
        std::cout << "The number is zero." << std::endl;
    }

    return 0;
```

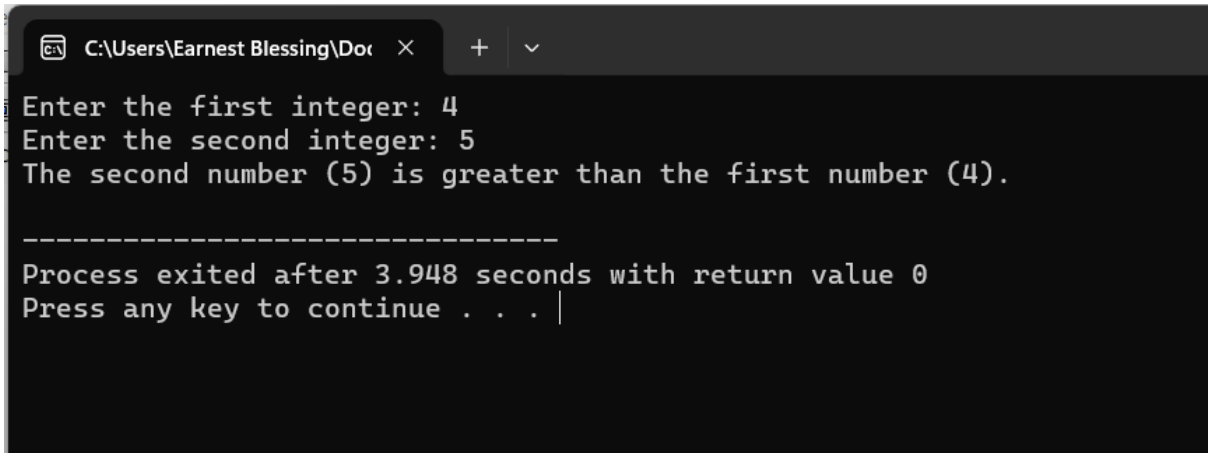



```
}
```

8. Program to determine which number is greater among two integers

```
#include <iostream>

int main() {
    // Declare variables
    int firstNumber, secondNumber;

    // Read two numbers from the user
    std::cout << "Enter the first integer: ";
    std::cin >> firstNumber;

    std::cout << "Enter the second integer: ";
```

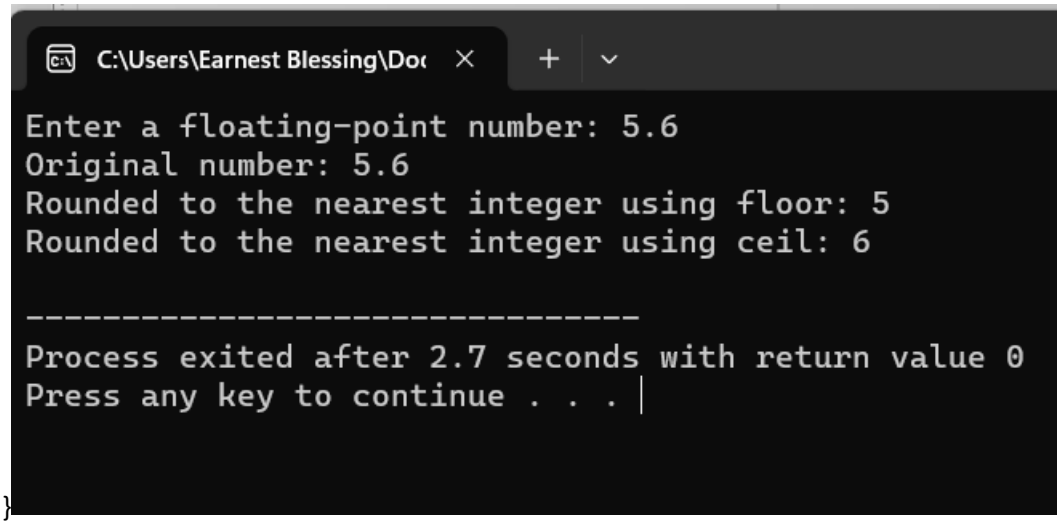
```
    std::cin >> secondNumber;


    // Determine the greater number
    if (firstNumber > secondNumber) {
        std::cout << "The first number (" << firstNumber << ") is greater than the second number (" << secondNumber << ")." << std::endl;
    } else if (secondNumber > firstNumber) {
        std::cout << "The second number (" << secondNumber << ") is greater than the first number (" << firstNumber << ")." << std::endl;
    } else {
        std::cout << "Both numbers are equal." << std::endl;
    }


    return 0;
}
```



9. Program to read a floating-number and round it to the nearest integer using the floor an ceil functions.

```
#include <iostream>
#include <cmath>


int main() {
    // Declare variables
    double floatingNumber;
```

```
    // Read a floating-point number from the user
    std::cout << "Enter a floating-point number: ";
    std::cin >> floatingNumber;

    // Round the number using floor and ceil functions
    int roundedFloor = static_cast<int>(std::floor(floatingNumber));
    int roundedCeil = static_cast<int>(std::ceil(floatingNumber));

    // Display the results
    std::cout << "Original number: " << floatingNumber << std::endl;
    std::cout << "Rounded to the nearest integer using floor: " << roundedFloor << std::endl;
    std::cout << "Rounded to the nearest integer using ceil: " << roundedCeil << std::endl;

    return 0;
```

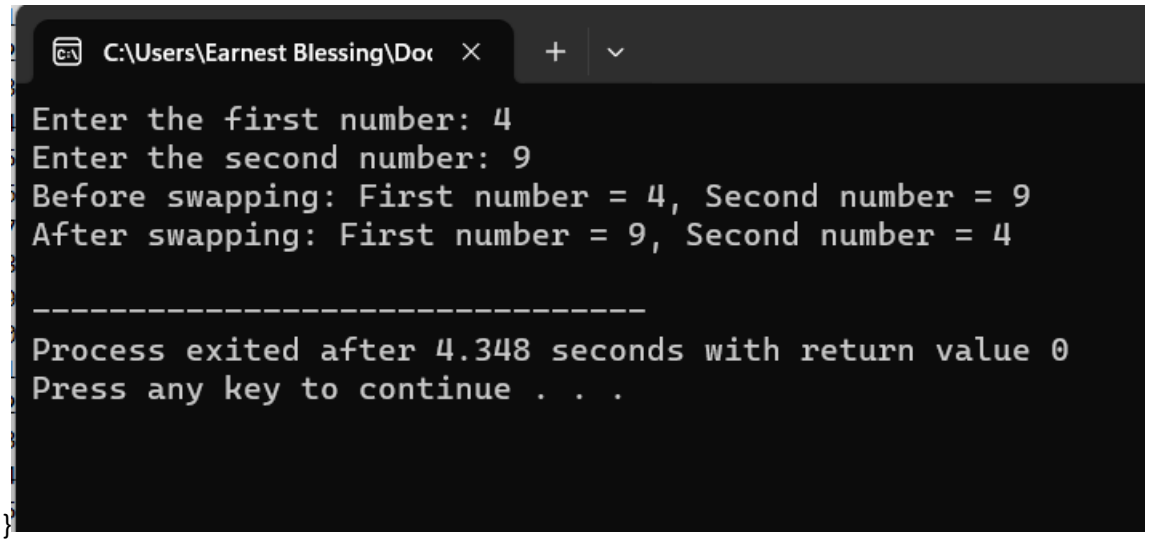



```
}
```

10. Program to swap two numbers using bitwise XOR operator

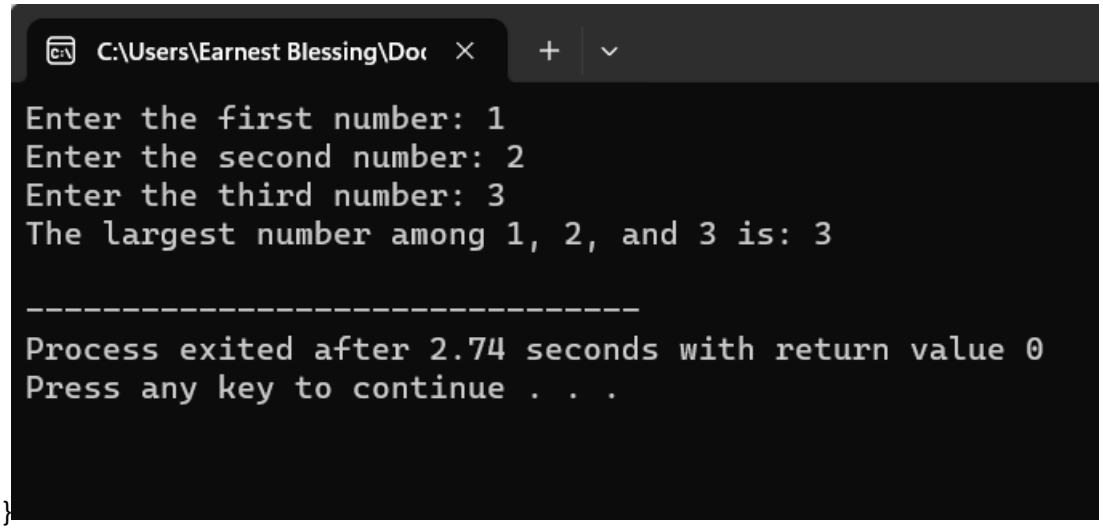
```
#include <iostream>

int main() {
    // Declare variables
    int firstNumber, secondNumber;
```

```
    // Read two numbers from the user
    std::cout << "Enter the first number: ";
    std::cin >> firstNumber;

    std::cout << "Enter the second number: ";
    std::cin >> secondNumber;

    // Display the numbers before swapping
    std::cout << "Before swapping: First number = " << firstNumber << ", Second number = " <<
secondNumber << std::endl;

    // Swap the numbers using bitwise XOR
    firstNumber = firstNumber ^ secondNumber;
    secondNumber = firstNumber ^ secondNumber;
    firstNumber = firstNumber ^ secondNumber;

    // Display the numbers after swapping
    std::cout << "After swapping: First number = " << firstNumber << ", Second number = " <<
secondNumber << std::endl;

    return 0;
}
```



11. Largest among three numbers using ternary conditional operator

```
#include <iostream>

int main() {
    // Declare variables
    int num1, num2, num3, largest;

    // Read three numbers from the user
    std::cout << "Enter the first number: ";
    std::cin >> num1;

    std::cout << "Enter the second number: ";
    std::cin >> num2;

    std::cout << "Enter the third number: ";
    std::cin >> num3;

    // Use ternary conditional operator to find the largest number
    largest = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);

    // Display the largest number
    std::cout << "The largest number among " << num1 << ", " << num2 << ", and " << num3 << " is: "
<< largest << std::endl;

    return 0;
```



}

12. Program to check two numbers are equal or not using ternary conditional operator

```
#include <iostream>


int main() {
    // Declare variables
    int num1, num2;

    // Read two numbers from the user
    std::cout << "Enter the first number: ";
    std::cin >> num1;

    std::cout << "Enter the second number: ";
    std::cin >> num2;

    // Use ternary conditional operator to check equality
    std::cout << "The numbers are " << ((num1 == num2) ? "equal" : "not equal") << std::endl;

    return 0;
```

```
C:\Users\Earnest Blessing\Doc
Enter the first number: 4
Enter the second number: 5
The numbers are not equal

--------------------------------
Process exited after 3.744 seconds with return v
Press any key to continue . . .
```

13. Program to check the integer is divisible by 3 or not using ternary conditional operator

```
#include <iostream>


int main() {

   // Declare variable

   int number;


   // Read an integer from the user

   std::cout << "Enter an integer: ";

   std::cin >> number;


   // Use ternary conditional operator to check divisibility by 3

   std::cout << "The number is " << ((number % 3 == 0) ? "divisible by 3" : "not divisible by 3") <<
std::endl;


   return 0;
```

```
C:\Users\Earnest Blessing\Doc

Enter an integer: 21
The number is divisible by 3

--------------------------------
Process exited after 2.46 seconds with return value 0
Press any key to continue . . .
```

}

14. Program to print numbers from 1 to 10 using for loop

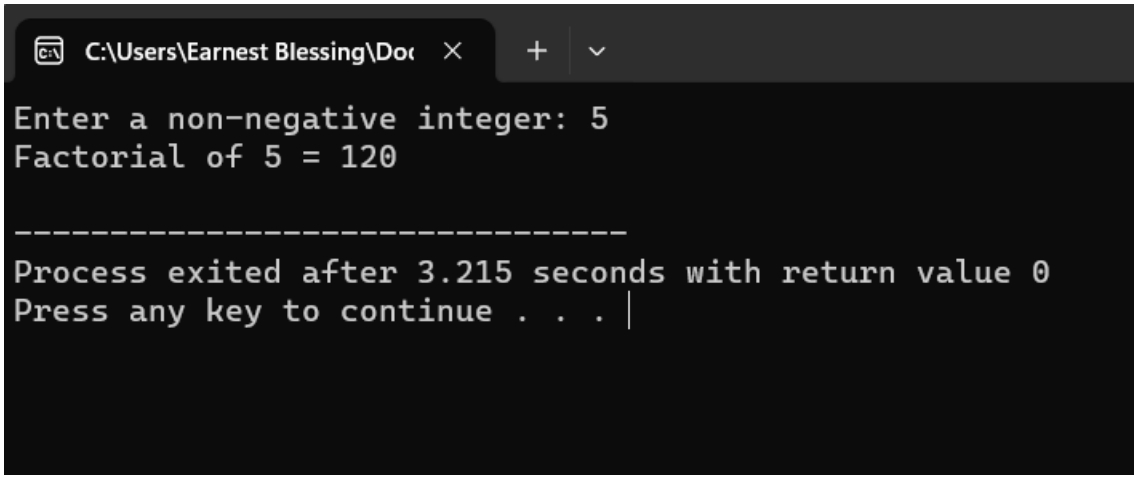
```
#include <iostream>


int main() {
   // Use a for loop to print numbers from 1 to 10
   for (int i = 1; i <= 10; ++i) {
      std::cout << i << " ";
   }

   // Add a newline at the end for better formatting
   std::cout << std::endl;

   return 0;
```

```
C:\Users\Earnest Blessing\Doc

1 2 3 4 5 6 7 8 9 10

--------------------------------
Process exited after 0.04859 seconds with return value 0
Press any key to continue . . .
```

}

15. Factorial of a number using for loop

```
#include <iostream>

int main() {
    // Declare variables
    int number;
    unsigned long long factorial = 1;  // Use unsigned long long to handle larger factorials

    // Read a number from the user
    std::cout << "Enter a non-negative integer: ";
    std::cin >> number;

    // Check if the number is non-negative
    if (number < 0) {
        std::cout << "Factorial is not defined for negative numbers." << std::endl;
    } else {
        // Calculate the factorial using a for loop
        for (int i = 1; i <= number; ++i) {
            factorial *= i;
        }

        // Display the factorial
        std::cout << "Factorial of " << number << " = " << factorial << std::endl;
    }

    return 0;
}
```



16.

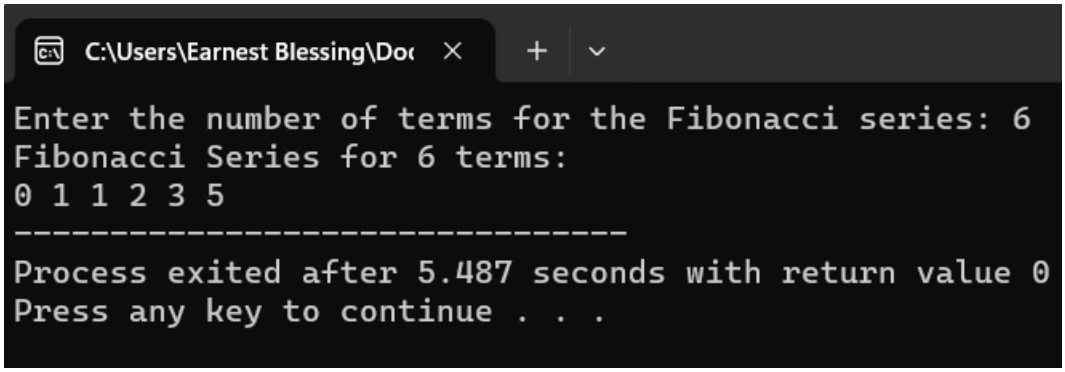
```
#include <iostream>

int main() {
   // Declare a variable to store the number for which the multiplication table will be printed
   int number;

   // Get the number from the user
   std::cout << "Enter a number to print its multiplication table: ";
   std::cin >> number;

   // Print the multiplication table using a for loop
   std::cout << "Multiplication Table for " << number << ":" << std::endl;
   for (int i = 1; i <= 10; ++i) {
      std::cout << number << " * " << i << " = " << (number * i) << std::endl;
   }

   return 0;
}
```

```
C:\Users\Earnest Blessing\Doc
Enter a number to print its multiplication table: 5
Multiplication Table for 5:
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

--------------------------------
Process exited after 4.725 seconds with return value 0
Press any key to continue . . .
```

17.

```
#include <iostream>

int main() {
    // Declare variables to store the Fibonacci series terms
    int n;

    // Get the number of terms in the Fibonacci series from the user
    std::cout << "Enter the number of terms for the Fibonacci series: ";
    std::cin >> n;

    // Initialize the first two terms of the Fibonacci series
    int firstTerm = 0, secondTerm = 1;

    // Print the Fibonacci series using a for loop
    std::cout << "Fibonacci Series for " << n << " terms:" << std::endl;
```

```
    for (int i = 0; i < n; ++i) {
        std::cout << firstTerm << " ";

        // Calculate the next term in the series
        int nextTerm = firstTerm + secondTerm;

        // Update firstTerm and secondTerm for the next iteration
        firstTerm = secondTerm;
        secondTerm = nextTerm;
    }

    return 0;
}
```



18.

```
#include <iostream>

int main() {
    // Declare variables
    int number;
    bool isPrime = true;
```

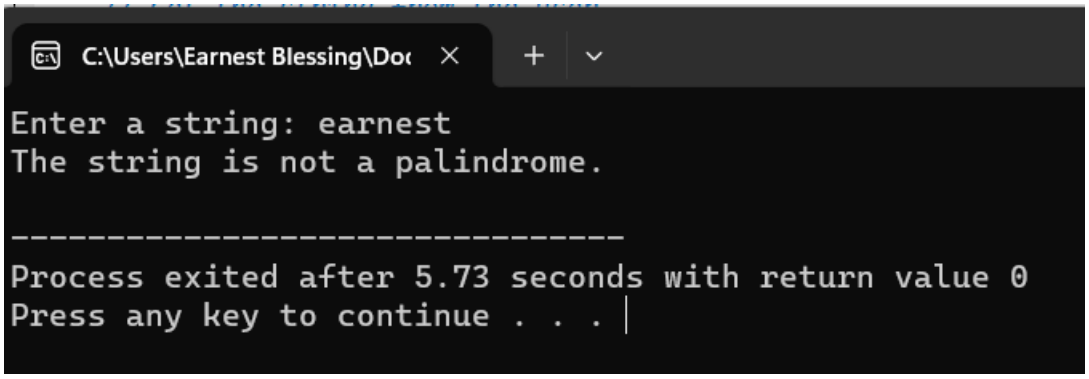
```
    // Get the number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> number;

    // Check if the number is greater than 1
    if (number > 1) {
        // Check for factors using a for loop
        for (int i = 2; i <= number / 2; ++i) {
            if (number % i == 0) {
                isPrime = false;
                break; // No need to check further if a factor is found
            }
        }
    } else {
        isPrime = false; // Numbers less than or equal to 1 are not prime
    }

    // Display the result
    if (isPrime) {
        std::cout << number << " is a prime number." << std::endl;
    } else {
        std::cout << number << " is not a prime number." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc
Enter a positive integer: 4
4 is not a prime number.

--------------------------------
Process exited after 14.64 seconds with return value 0
Press any key to continue . . .
```

19.

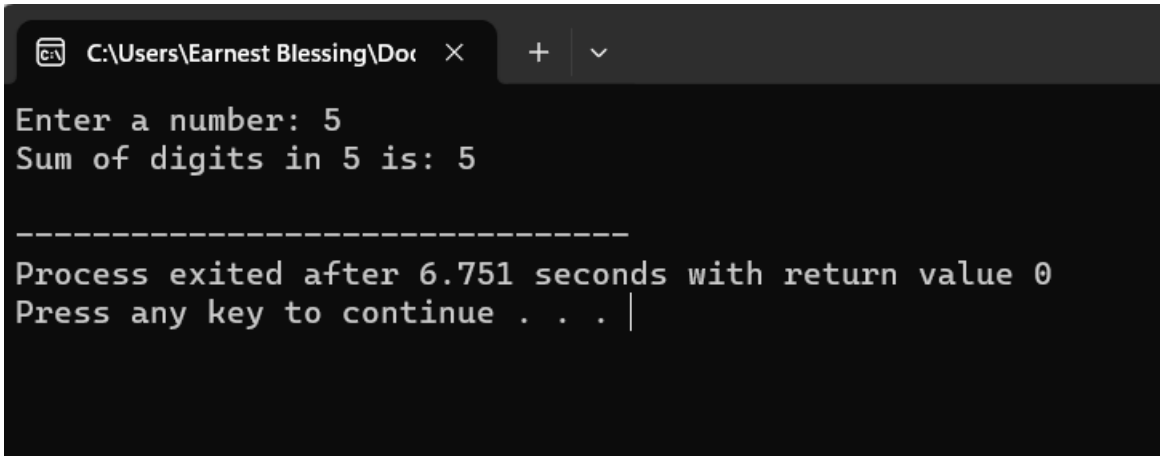
```
#include <iostream>
#include <cctype> // For using std::tolower

int main() {
    // Declare variables
    std::string inputString;

    // Get the string from the user
    std::cout << "Enter a string: ";
    std::getline(std::cin, inputString);

    // Remove spaces and convert the string to lowercase
    std::string processedString;
    for (char character : inputString) {
        if (!std::isspace(character)) {
            processedString += std::tolower(character);
        }
    }

    // Check if the processed string is a palindrome using a while loop
```

```
    bool isPalindrome = true;
    int start = 0;
    int end = processedString.length() - 1;

    while (start < end) {
        if (processedString[start] != processedString[end]) {
            isPalindrome = false;
            break;
        }
        ++start;
        --end;
    }

    // Display the result
    if (isPalindrome) {
        std::cout << "The string is a palindrome." << std::endl;
    } else {
        std::cout << "The string is not a palindrome." << std::endl;
    }

    return 0;
}
```



20.

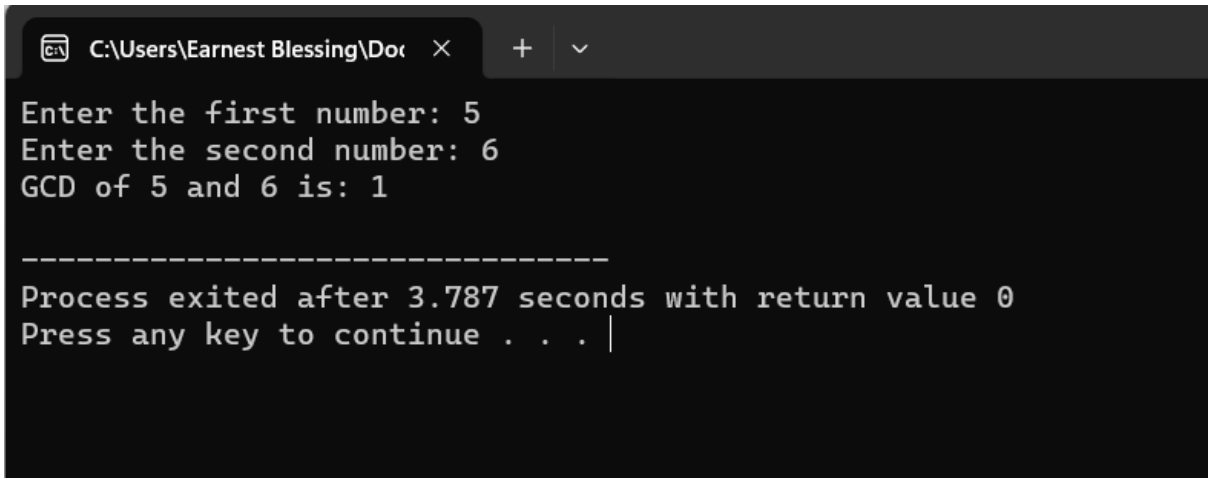
```
#include <iostream>

int main() {
    // Declare variables
    int number, originalNumber;
    int sum = 0;

    // Get the number from the user
    std::cout << "Enter a number: ";
    std::cin >> number;

    // Save the original number for later comparison
    originalNumber = number;

    // Calculate the sum of digits using a while loop
    while (number != 0) {
        // Extract the last digit
        int digit = number % 10;

        // Add the digit to the sum
        sum += digit;

        // Remove the last digit
        number /= 10;
    }

    // Display the result
    std::cout << "Sum of digits in " << originalNumber << " is: " << sum << std::endl;

    return 0;
}
```



21.

```
#include <iostream>

// Function to calculate GCD using Euclidean Algorithm
int calculateGCD(int a, int b) {
    do {
        int temp = a;
        a = b;
        b = temp % b;
    } while (b != 0);

    return a;
}

int main() {
    // Declare variables
    int num1, num2;

    // Get two numbers from the user
    std::cout << "Enter the first number: ";
    std::cin >> num1;
```

```
    std::cout << "Enter the second number: ";
    std::cin >> num2;

    // Calculate and display the GCD using the calculateGCD function
    int gcd = calculateGCD(num1, num2);
    std::cout << "GCD of " << num1 << " and " << num2 << " is: " << gcd << std::endl;

    return 0;
}
```



22.

```
#include <iostream>

// Function to check if a number is perfect
bool isPerfectNumber(int number) {
    int sum = 1; // 1 is always a divisor

    for (int i = 2; i <= number / 2; ++i) {
        if (number % i == 0) {
            sum += i;
        }
    }
```

```
    return (sum == number);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is perfect
    if (num > 0) {
        if (isPerfectNumber(num)) {
            std::cout << num << " is a perfect number." << std::endl;
        } else {
            std::cout << num << " is not a perfect number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc
Enter a positive integer: 5
5 is not a perfect number.

--------------------------------
Process exited after 2.477 seconds with return value 0
Press any key to continue . . .
```

23.

```
#include <iostream>
#include <cmath>

// Function to calculate the number of digits in a number
int countDigits(int number) {
    int count = 0;
    while (number != 0) {
        number /= 10;
        ++count;
    }
    return count;
}

// Function to check if a number is an Armstrong number
bool isArmstrongNumber(int number) {
    int originalNumber = number;
    int numDigits = countDigits(number);
    int sum = 0;

    while (number != 0) {
        int digit = number % 10;
```

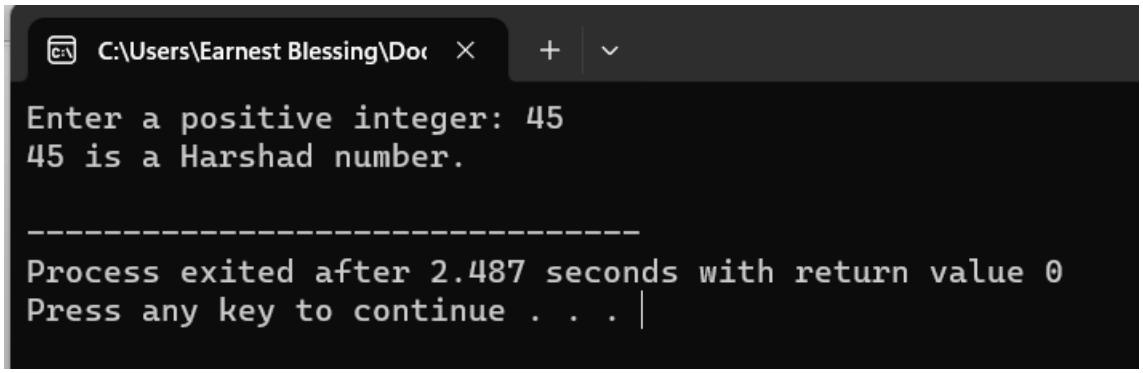
```
        sum += static_cast<int>(std::pow(digit, numDigits));
        number /= 10;
    }

    return (sum == originalNumber);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a number: ";
    std::cin >> num;

    // Check and display if the number is an Armstrong number
    if (num >= 0) {
        if (isArmstrongNumber(num)) {
            std::cout << num << " is an Armstrong number." << std::endl;
        } else {
            std::cout << num << " is not an Armstrong number." << std::endl;
        }
    } else {
        std::cout << "Please enter a non-negative integer." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc
Enter a number: 131
131 is not an Armstrong number.

--------------------------------
Process exited after 4.775 seconds with return value 0
Press any key to continue . . .
```

24.

```
#include <iostream>

// Function to calculate the sum of digits in a number
int sumOfDigits(int number) {
    int sum = 0;
    while (number != 0) {
        sum += number % 10;
        number /= 10;
    }
    return sum;
}

// Function to check if a number is a Harshad number
bool isHarshadNumber(int number) {
    int digitSum = sumOfDigits(number);
    return (number % digitSum == 0);
}

int main() {
    // Declare variables
```

```
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is a Harshad number
    if (num > 0) {
        if (isHarshadNumber(num)) {
            std::cout << num << " is a Harshad number." << std::endl;
        } else {
            std::cout << num << " is not a Harshad number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```

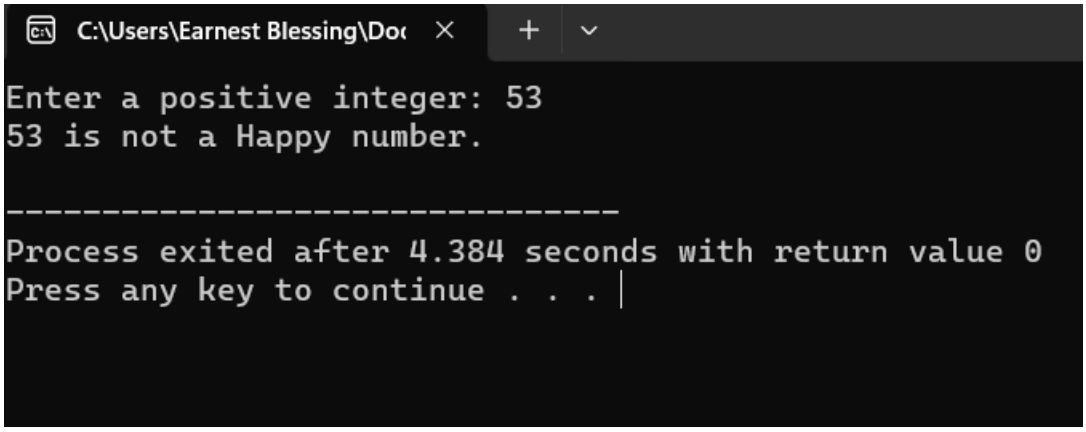


25.

```
#include <iostream>
#include <unordered_set>
```

```
// Function to calculate the sum of squares of digits in a number
int sumOfSquares(int number) {
    int sum = 0;
    while (number != 0) {
        int digit = number % 10;
        sum += digit * digit;
        number /= 10;
    }
    return sum;
}

// Function to check if a number is a Happy number
bool isHappyNumber(int number) {
    std::unordered_set<int> visitedNumbers;

    while (number != 1 && visitedNumbers.find(number) == visitedNumbers.end()) {
        visitedNumbers.insert(number);
        number = sumOfSquares(number);
    }

    return (number == 1);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;
```

```
    // Check and display if the number is a Happy number
    if (num > 0) {
        if (isHappyNumber(num)) {
            std::cout << num << " is a Happy number." << std::endl;
        } else {
            std::cout << num << " is not a Happy number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```

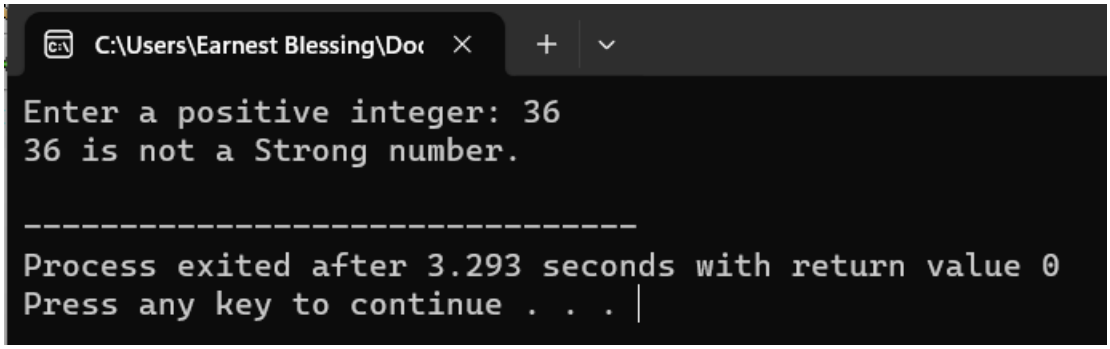


26.

```
#include <iostream>

// Function to calculate the factorial of a number
int factorial(int num) {
    if (num == 0 || num == 1) {
        return 1;
```

```
    }
    return num * factorial(num - 1);
}

// Function to calculate the sum of factorials of digits in a number
int sumOfFactorials(int number) {
    int sum = 0;
    int originalNumber = number;

    while (number != 0) {
        int digit = number % 10;
        sum += factorial(digit);
        number /= 10;
    }

    return sum;
}

// Function to check if a number is a Strong number
bool isStrongNumber(int number) {
    return (sumOfFactorials(number) == number);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;
```

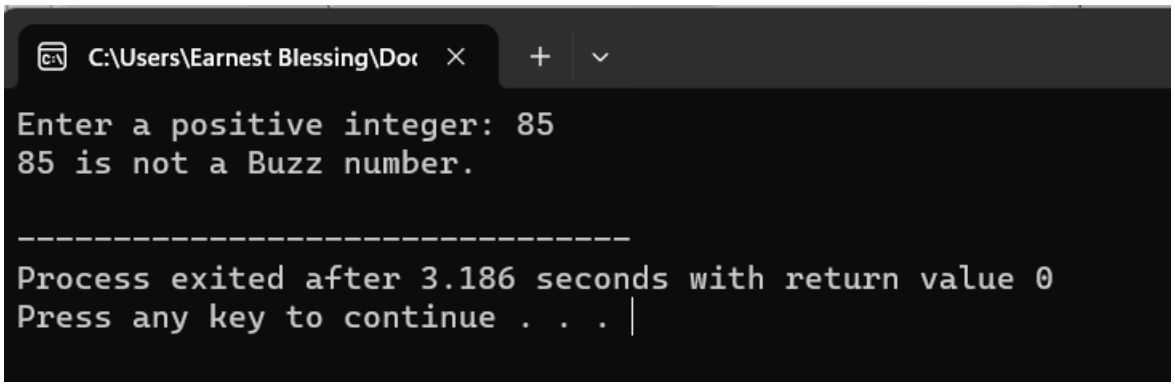
```
    // Check and display if the number is a Strong number
    if (num >= 0) {
        if (isStrongNumber(num)) {
            std::cout << num << " is a Strong number." << std::endl;
        } else {
            std::cout << num << " is not a Strong number." << std::endl;
        }
    } else {
        std::cout << "Please enter a non-negative integer." << std::endl;
    }

    return 0;
}
```



27.

```
#include <iostream>

// Function to check if a number is a Buzz number
bool isBuzzNumber(int number) {
    return (number % 7 == 0) || (number % 10 == 7);
}

int main() {
    // Declare variables
```

```
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is a Buzz number
    if (num > 0) {
        if (isBuzzNumber(num)) {
            std::cout << num << " is a Buzz number." << std::endl;
        } else {
            std::cout << num << " is not a Buzz number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```



27.

```
#include <iostream>
```

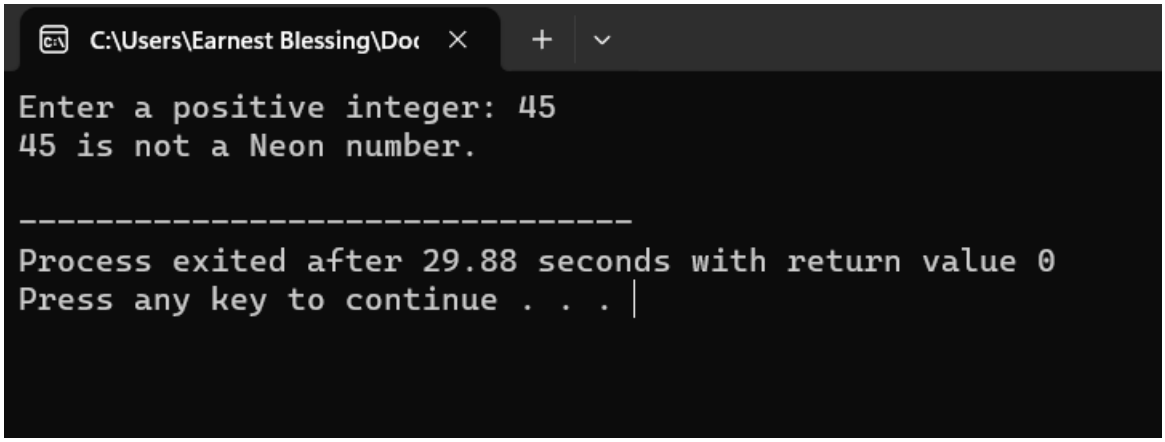
```
// Function to check if a number is a Neon number
bool isNeonNumber(int number) {
    int square = number * number;
    int digitSum = 0;

    while (square != 0) {
        digitSum += square % 10;
        square /= 10;
    }

    return (digitSum == number);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is a Neon number
    if (num > 0) {
        if (isNeonNumber(num)) {
            std::cout << num << " is a Neon number." << std::endl;
        } else {
            std::cout << num << " is not a Neon number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }
```

```
    return 0;
}
```

28.

```
#include <iostream>

// Function to check if a number is a Neon number
bool isNeonNumber(int number) {
    int square = number * number;
    int digitSum = 0;

    while (square != 0) {
        digitSum += square % 10;
        square /= 10;
    }

    return (digitSum == number);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is a Neon number
    if (num > 0) {
```

```
        if (isNeonNumber(num)) {
            std::cout << num << " is a Neon number." << std::endl;
        } else {
            std::cout << num << " is not a Neon number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```



29.

```
#include <iostream>

// Function to calculate the sum of proper divisors of a number
int sumOfDivisors(int number) {
    int sum = 1; // 1 is always a divisor

    for (int i = 2; i <= number / 2; ++i) {
        if (number % i == 0) {
            sum += i;
```

```
        }
    }

    return sum;
}

// Function to check if a number is an abundant number
bool isAbundantNumber(int number) {
    return (sumOfDivisors(number) > number);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is an abundant number
    if (num > 0) {
        if (isAbundantNumber(num)) {
            std::cout << num << " is an abundant number." << std::endl;
        } else {
            std::cout << num << " is not an abundant number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
```

}

```
C:\Users\Earnest Blessing\Doc

Enter a positive integer: 89
89 is not an abundant number.

--------------------------------
Process exited after 2.901 seconds with return value 0
Press any key to continue . . .
```

30.

```
#include <iostream>
#include <cmath>

// Function to calculate the number of digits in a number
int countDigits(int number) {
    int count = 0;
    while (number != 0) {
        number /= 10;
        ++count;
    }
    return count;
}

// Function to check if a number is a narcissistic number
bool isNarcissisticNumber(int number) {
    int originalNumber = number;
    int numDigits = countDigits(number);
    int sum = 0;

    while (number != 0) {
```

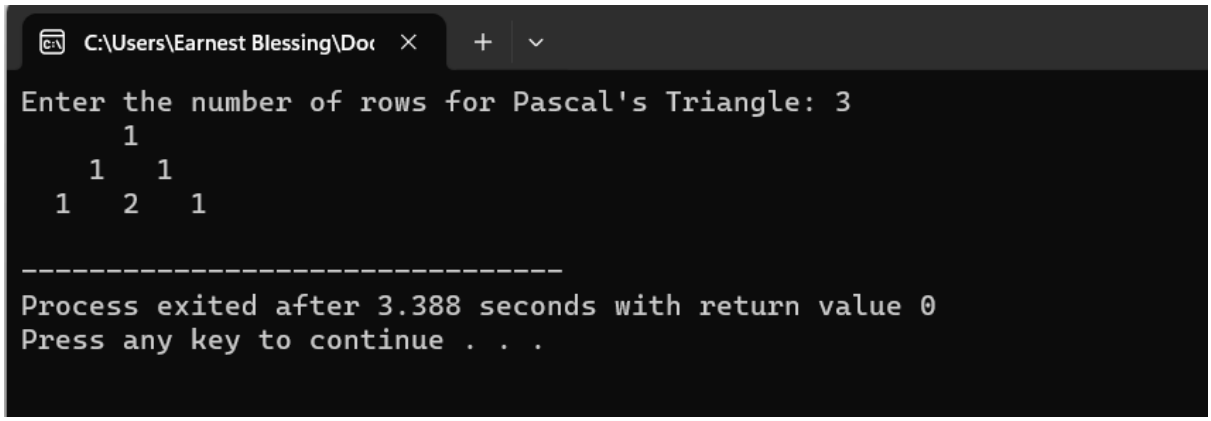
```
        int digit = number % 10;
        sum += static_cast<int>(std::pow(digit, numDigits));
        number /= 10;
    }

    return (sum == originalNumber);
}

int main() {
    // Declare variables
    int num;

    // Get a number from the user
    std::cout << "Enter a positive integer: ";
    std::cin >> num;

    // Check and display if the number is a narcissistic number
    if (num > 0) {
        if (isNarcissisticNumber(num)) {
            std::cout << num << " is a narcissistic number." << std::endl;
        } else {
            std::cout << num << " is not a narcissistic number." << std::endl;
        }
    } else {
        std::cout << "Please enter a positive integer." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc

Enter a positive integer: 121
121 is not a narcissistic number.

--------------------------------
Process exited after 15.72 seconds with return value 0
Press any key to continue . . .
```

33.

```
#include <iostream>

// Function to calculate and print Pascal's Triangle
void printPascalsTriangle(int rows) {
    for (int i = 0; i < rows; ++i) {
        int coefficient = 1;

        // Print leading spaces for each row
        for (int space = 1; space <= rows - i; ++space) {
            std::cout << "  ";
        }

        for (int j = 0; j <= i; ++j) {
            // Calculate the next coefficient
            if (j > 0) {
                coefficient = coefficient * (i - j + 1) / j;
            }

            // Print the coefficient
            std::cout << coefficient << "   ";
        }
```

```
        std::cout << std::endl;
    }
}

int main() {
    // Declare variables
    int numRows;

    // Get the number of rows from the user
    std::cout << "Enter the number of rows for Pascal's Triangle: ";
    std::cin >> numRows;

    // Check and display Pascal's Triangle
    if (numRows > 0) {
        printPascalsTriangle(numRows);
    } else {
        std::cout << "Please enter a positive integer for the number of rows." << std::endl;
    }

    return 0;
}
```



34.

```
#include <iostream>

// Function to print a diamond pattern with asterisks
void printDiamond(int n) {
    // Print upper part of the diamond
    for (int i = 1; i <= n; ++i) {
        // Print leading spaces
        for (int space = 1; space <= n - i; ++space) {
            std::cout << " ";
        }

        // Print asterisks
        for (int j = 1; j <= 2 * i - 1; ++j) {
            std::cout << "*";
        }

        std::cout << std::endl;
    }

    // Print lower part of the diamond
    for (int i = n - 1; i >= 1; --i) {
        // Print leading spaces
        for (int space = 1; space <= n - i; ++space) {
            std::cout << " ";
        }

        // Print asterisks
        for (int j = 1; j <= 2 * i - 1; ++j) {
            std::cout << "*";
        }
```

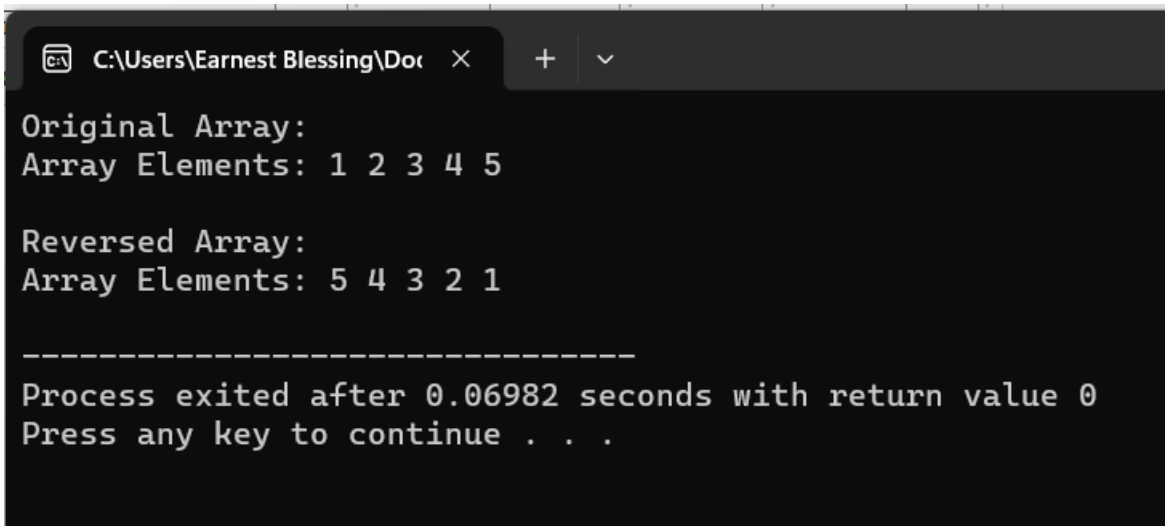
```
        std::cout << std::endl;
    }
}

int main() {
    // Declare variable
    int numRows;

    // Get the number of rows for the diamond from the user
    std::cout << "Enter the number of rows for the diamond: ";
    std::cin >> numRows;

    // Check and display the diamond pattern
    if (numRows > 0) {
        printDiamond(numRows);
    } else {
        std::cout << "Please enter a positive integer for the number of rows." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc

Enter the number of rows for the diamond: 5
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *

--------------------------------
Process exited after 3.079 seconds with return value 0
Press any key to continue . . .
```

35.

```
#include <iostream>

// Function to reverse the elements in an array
void reverseArray(int arr[], int size) {
    for (int i = 0; i < size / 2; ++i) {
        // Swap elements at the beginning and end of the array
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
```

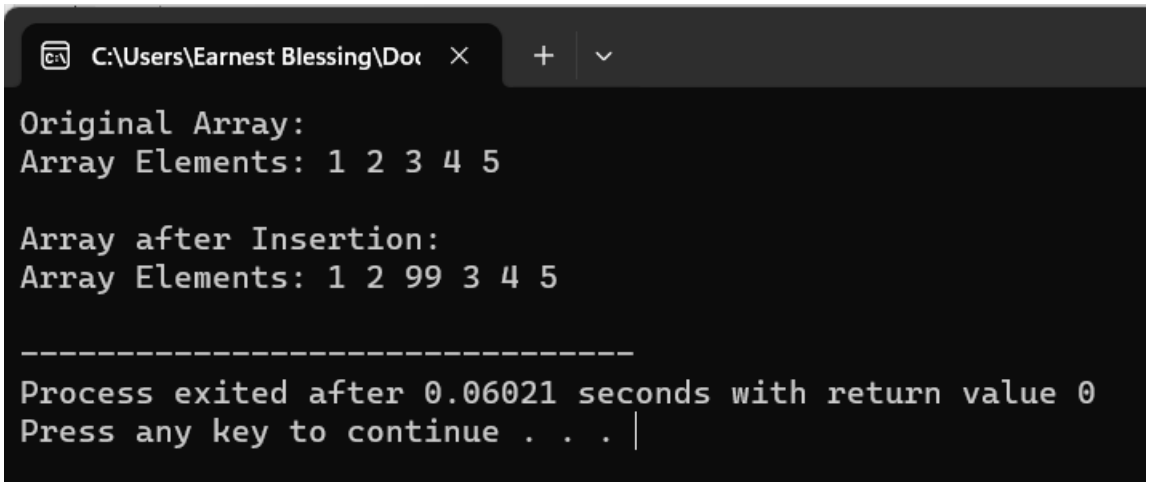
```
    }
    std::cout << std::endl;
}

int main() {
    const int size = 5;
    int myArray[size] = {1, 2, 3, 4, 5};

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, size);

    // Reverse the elements in the array
    reverseArray(myArray, size);

    // Display the reversed array
    std::cout << "\nReversed Array:" << std::endl;
    displayArray(myArray, size);

    return 0;
}
```



36.

```
#include <iostream>

// Function to insert an element in an array at a specific position
void insertElement(int arr[], int& size, int position, int element) {
    // Check if the position is valid
    if (position >= 0 && position <= size) {
        // Shift elements to make space for the new element
        for (int i = size; i > position; --i) {
            arr[i] = arr[i - 1];
        }

        // Insert the new element at the specified position
        arr[position] = element;

        // Increase the size of the array
        ++size;
    } else {
        std::cout << "Invalid position. Element not inserted." << std::endl;
    }
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}
```

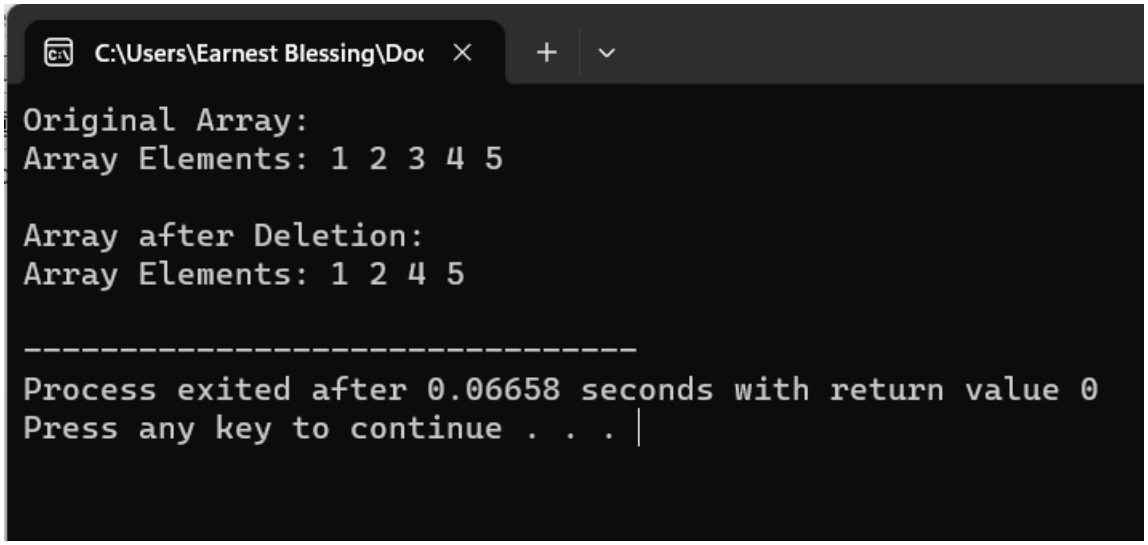
```
int main() {
    const int maxSize = 10; // Maximum size of the array
    int myArray[maxSize] = {1, 2, 3, 4, 5};
    int arraySize = 5;

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Insert an element (e.g., 99) at a specific position (e.g., position 2)
    int insertPosition = 2;
    int elementToInsert = 99;
    insertElement(myArray, arraySize, insertPosition, elementToInsert);

    // Display the array after insertion
    std::cout << "\nArray after Insertion:" << std::endl;
    displayArray(myArray, arraySize);

    return 0;
}
```



37.

```
#include <iostream>

// Function to delete an element in an array at a specific position
void deleteElement(int arr[], int& size, int position) {
    // Check if the position is valid
    if (position >= 0 && position < size) {
        // Shift elements to fill the gap left by the deleted element
        for (int i = position; i < size - 1; ++i) {
            arr[i] = arr[i + 1];
        }

        // Decrease the size of the array
        --size;
    } else {
        std::cout << "Invalid position. Element not deleted." << std::endl;
    }
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 10; // Maximum size of the array
    int myArray[maxSize] = {1, 2, 3, 4, 5};
    int arraySize = 5;
```

```
    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Delete an element at a specific position (e.g., position 2)
    int deletePosition = 2;
    deleteElement(myArray, arraySize, deletePosition);

    // Display the array after deletion
    std::cout << "\nArray after Deletion:" << std::endl;
    displayArray(myArray, arraySize);

    return 0;
}
```



38.

```
#include <iostream>

// Function to calculate the sum of elements in an array
int calculateSum(const int arr[], int size) {
```

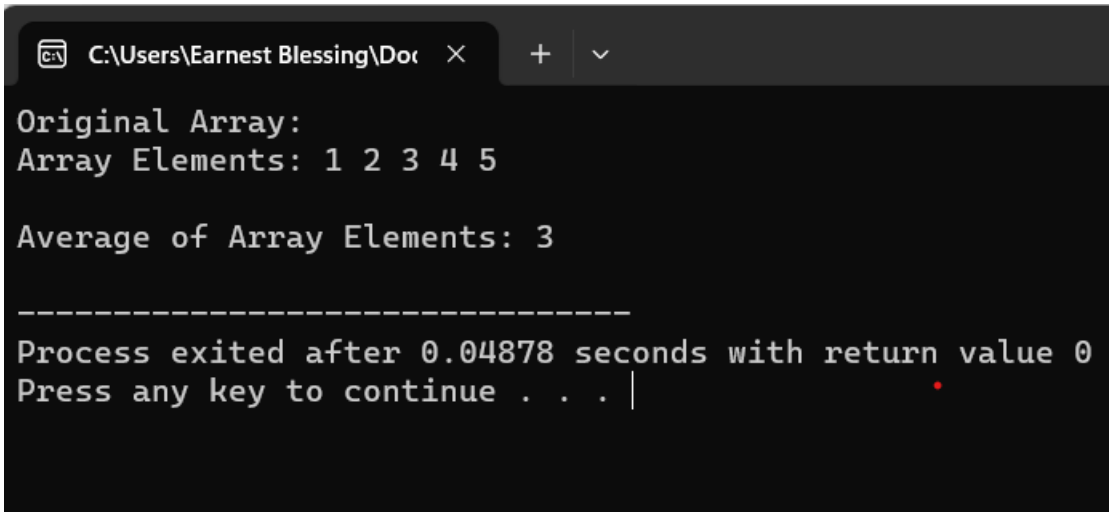
```
    int sum = 0;
    for (int i = 0; i < size; ++i) {
        sum += arr[i];
    }
    return sum;
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 5; // Maximum size of the array
    int myArray[maxSize] = {1, 2, 3, 4, 5};
    int arraySize = 5;

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Calculate and display the sum of elements in the array
    int sum = calculateSum(myArray, arraySize);
    std::cout << "\nSum of Array Elements: " << sum << std::endl;

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc
Original Array:
Array Elements: 1 2 3 4 5

Sum of Array Elements: 15

--------------------------------
Process exited after 0.06014 seconds with return value 0
Press any key to continue . . .
```

39.

```
#include <iostream>

// Function to calculate the average of elements in an array
double calculateAverage(const int arr[], int size) {
    if (size == 0) {
        return 0.0; // To avoid division by zero
    }

    int sum = 0;
    for (int i = 0; i < size; ++i) {
        sum += arr[i];
    }

    return static_cast<double>(sum) / size;
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
```

```
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 5; // Maximum size of the array
    int myArray[maxSize] = {1, 2, 3, 4, 5};
    int arraySize = 5;

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Calculate and display the average of elements in the array
    double average = calculateAverage(myArray, arraySize);
    std::cout << "\nAverage of Array Elements: " << average << std::endl;

    return 0;
}
```



40.

```
#include <iostream>
#include <climits>

// Function to find the second largest element in an array
int findSecondLargest(const int arr[], int size) {
    if (size < 2) {
        std::cerr << "Array should have at least two elements." << std::endl;
        return INT_MIN; // Return the minimum integer value as an error indicator
    }

    int firstMax = arr[0];
    int secondMax = INT_MIN;

    for (int i = 1; i < size; ++i) {
        if (arr[i] > firstMax) {
            secondMax = firstMax;
            firstMax = arr[i];
        } else if (arr[i] > secondMax && arr[i] < firstMax) {
            secondMax = arr[i];
        }
    }

    return secondMax;
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
```

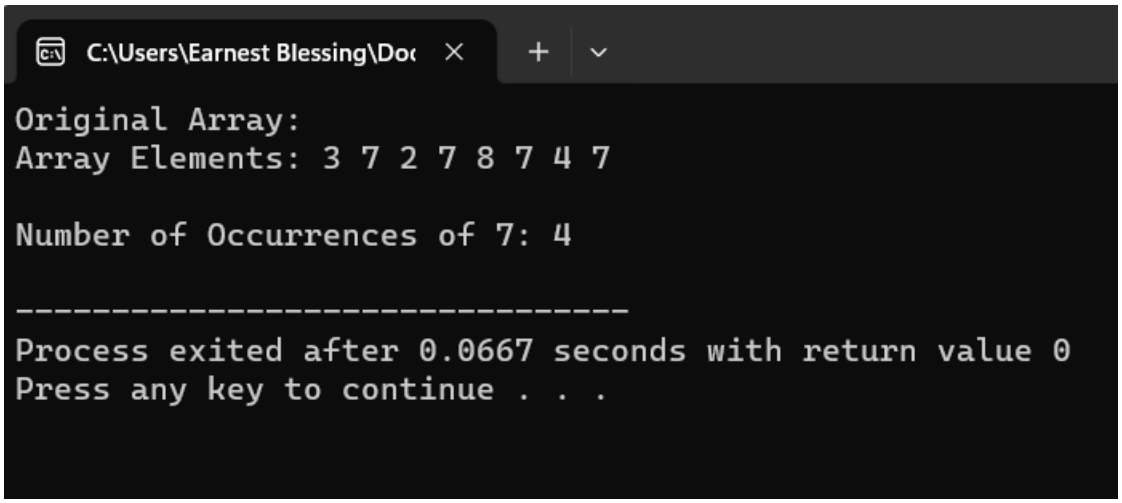
```
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize = 6; // Maximum size of the array
    int myArray[maxSize] = {7, 2, 3, 8, 5, 1};
    int arraySize = 6;

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Find and display the second largest element in the array
    int secondLargest = findSecondLargest(myArray, arraySize);
    if (secondLargest != INT_MIN) {
        std::cout << "\nSecond Largest Element: " << secondLargest << std::endl;
    } else {
        std::cout << "\nError finding second largest element." << std::endl;
    }

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc

Original Array:
Array Elements: 7 2 3 8 5 1

Second Largest Element: 7

--------------------------------
Process exited after 0.06038 seconds with return value 0
Press any key to continue . . .
```

41.

```
#include <iostream>

// Function to find the number of occurrences of a value in an array
int countOccurrences(const int arr[], int size, int targetValue) {
    int count = 0;
    for (int i = 0; i < size; ++i) {
        if (arr[i] == targetValue) {
            ++count;
        }
    }
    return count;
}

// Function to display the elements in an array
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
```

```
}

int main() {
    const int maxSize = 8; // Maximum size of the array
    int myArray[maxSize] = {3, 7, 2, 7, 8, 7, 4, 7};
    int arraySize = 8;
    int targetValue = 7; // Value to find occurrences

    // Display the original array
    std::cout << "Original Array:" << std::endl;
    displayArray(myArray, arraySize);

    // Find and display the number of occurrences of the target value in the array
    int occurrences = countOccurrences(myArray, arraySize, targetValue);
    std::cout << "\nNumber of Occurrences of " << targetValue << ": " << occurrences << std::endl;

    return 0;
}
```



42.

```
#include <iostream>

// Function to merge two arrays
void mergeArrays(const int arr1[], int size1, const int arr2[], int size2, int mergedArray[], int&
mergedSize) {
    int i = 0, j = 0, k = 0;

    // Merge arrays while there are elements in both arrays
    while (i < size1 && j < size2) {
        if (arr1[i] <= arr2[j]) {
            mergedArray[k++] = arr1[i++];
        } else {
            mergedArray[k++] = arr2[j++];
        }
    }

    // Copy the remaining elements from the first array, if any
    while (i < size1) {
        mergedArray[k++] = arr1[i++];
    }

    // Copy the remaining elements from the second array, if any
    while (j < size2) {
        mergedArray[k++] = arr2[j++];
    }

    // Set the merged size
    mergedSize = k;
}

// Function to display the elements in an array
```

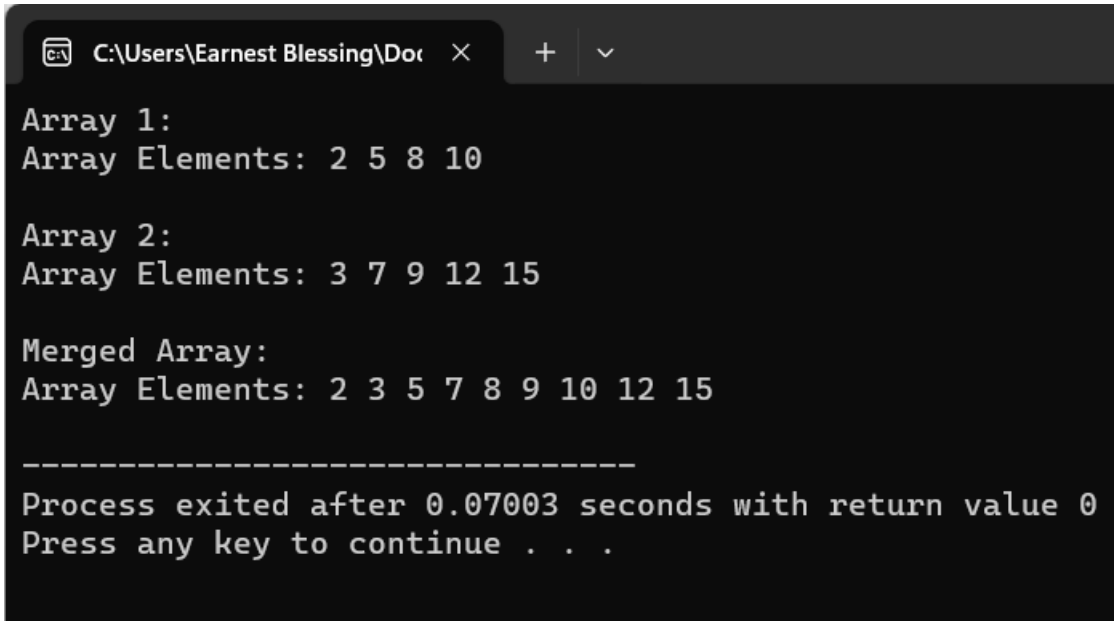
```
void displayArray(const int arr[], int size) {
    std::cout << "Array Elements: ";
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

int main() {
    const int maxSize1 = 4; // Maximum size of the first array
    const int maxSize2 = 5; // Maximum size of the second array
    int array1[maxSize1] = {2, 5, 8, 10};
    int array2[maxSize2] = {3, 7, 9, 12, 15};
    int mergedArray[maxSize1 + maxSize2];
    int mergedSize = 0;

    // Display the first array
    std::cout << "Array 1:" << std::endl;
    displayArray(array1, maxSize1);

    // Display the second array
    std::cout << "\nArray 2:" << std::endl;
    displayArray(array2, maxSize2);

    // Merge the arrays and display the result
    mergeArrays(array1, maxSize1, array2, maxSize2, mergedArray, mergedSize);
    std::cout << "\nMerged Array:" << std::endl;
    displayArray(mergedArray, mergedSize);

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc

Array 1:
Array Elements: 2 5 8 10

Array 2:
Array Elements: 3 7 9 12 15

Merged Array:
Array Elements: 2 3 5 7 8 9 10 12 15

--------------------------------
Process exited after 0.07003 seconds with return value 0
Press any key to continue . . .
```

43.

```
#include <iostream>

int main() {
    int size;

    // Get the size of the dynamic array from the user
    std::cout << "Enter the size of the dynamic array: ";
    std::cin >> size;

    // Create a dynamic array using pointers
    int* dynamicArray = new int[size];

    // Initialize the dynamic array with user input
    std::cout << "Enter " << size << " values for the dynamic array:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cin >> dynamicArray[i];
    }
```

```
    // Display the values in the dynamic array
    std::cout << "Values in the dynamic array:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cout << dynamicArray[i] << " ";
    }

    // Free the memory allocated for the dynamic array
    delete[] dynamicArray;

    return 0;
}
```

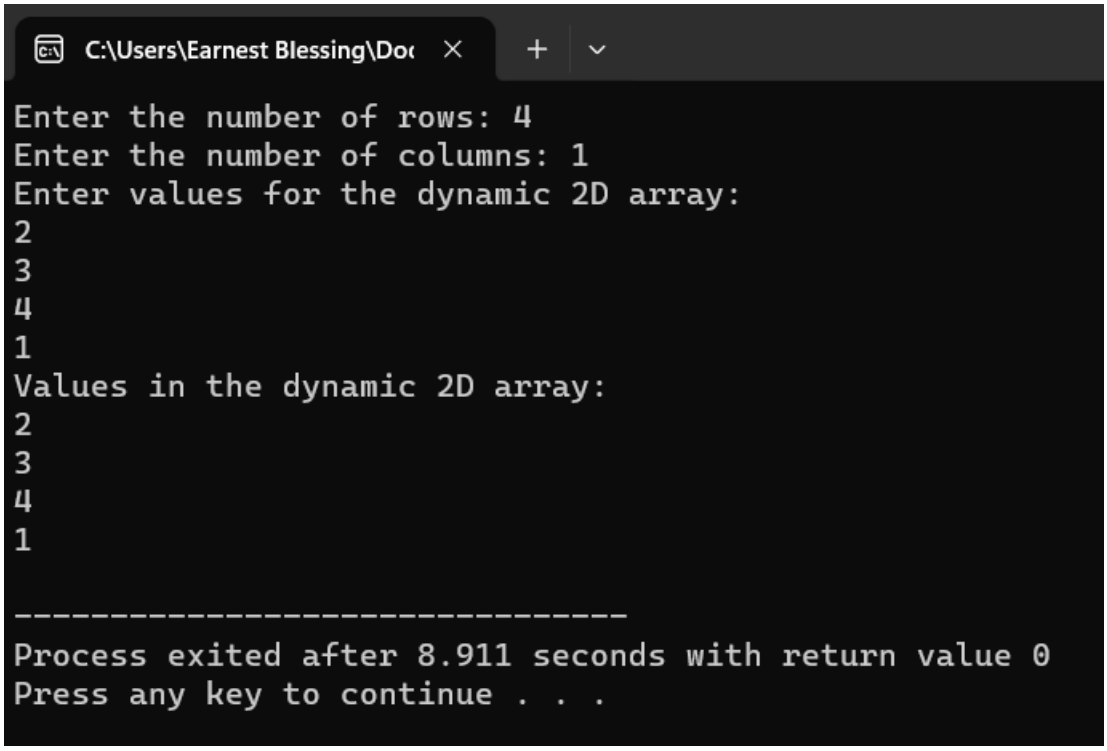


44.

```
#include <iostream>

int main() {
    int rows, cols;
```

```
// Get the number of rows and columns for the dynamic 2D array from the user
std::cout << "Enter the number of rows: ";
std::cin >> rows;
std::cout << "Enter the number of columns: ";
std::cin >> cols;

// Create a dynamic 2D array using pointers
int** dynamic2DArray = new int*[rows];
for (int i = 0; i < rows; ++i) {
    dynamic2DArray[i] = new int[cols];
}

// Initialize the dynamic 2D array with user input
std::cout << "Enter values for the dynamic 2D array:" << std::endl;
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        std::cin >> dynamic2DArray[i][j];
    }
}

// Display the values in the dynamic 2D array
std::cout << "Values in the dynamic 2D array:" << std::endl;
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        std::cout << dynamic2DArray[i][j] << " ";
    }
    std::cout << std::endl;
}

// Free the memory allocated for the dynamic 2D array
```

```
    for (int i = 0; i < rows; ++i) {
        delete[] dynamic2DArray[i];
    }
    delete[] dynamic2DArray;

    return 0;
}
```



45.

```
#include <iostream>

// Function to dynamically allocate memory for a matrix
int** allocateMatrix(int rows, int cols) {
    int** matrix = new int*[rows];
    for (int i = 0; i < rows; ++i) {
        matrix[i] = new int[cols];
    }
```

```
    return matrix;
}

// Function to deallocate memory for a matrix
void deallocateMatrix(int** matrix, int rows) {
    for (int i = 0; i < rows; ++i) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

// Function to add two matrices
void addMatrices(int** matrix1, int** matrix2, int** result, int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

// Function to display the elements in a matrix
void displayMatrix(int** matrix, int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
}

int main() {
```

```
int rows, cols;

// Get the number of rows and columns for the matrices from the user
std::cout << "Enter the number of rows: ";
std::cin >> rows;
std::cout << "Enter the number of columns: ";
std::cin >> cols;

// Allocate memory for matrices
int** matrix1 = allocateMatrix(rows, cols);
int** matrix2 = allocateMatrix(rows, cols);
int** resultMatrix = allocateMatrix(rows, cols);

// Input values for the matrices
std::cout << "Enter values for Matrix 1:" << std::endl;
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        std::cin >> matrix1[i][j];
    }
}

std::cout << "Enter values for Matrix 2:" << std::endl;
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        std::cin >> matrix2[i][j];
    }
}

// Add matrices
addMatrices(matrix1, matrix2, resultMatrix, rows, cols);
```

```
    // Display matrices and the result
    std::cout << "\nMatrix 1:" << std::endl;
    displayMatrix(matrix1, rows, cols);

    std::cout << "\nMatrix 2:" << std::endl;
    displayMatrix(matrix2, rows, cols);

    std::cout << "\nSum of Matrices:" << std::endl;
    displayMatrix(resultMatrix, rows, cols);

    // Deallocate memory for matrices
    deallocateMatrix(matrix1, rows);
    deallocateMatrix(matrix2, rows);
    deallocateMatrix(resultMatrix, rows);

    return 0;
}
```

```
C:\Users\Earnest Blessing\Doc

Enter the number of rows: 4
Enter the number of columns: 1
Enter values for the dynamic 2D array:
2
2
3
5
Values in the dynamic 2D array:
2
2
3
5

--------------------------------
Process exited after 8.179 seconds with return value 0
Press any key to continue . . .
```

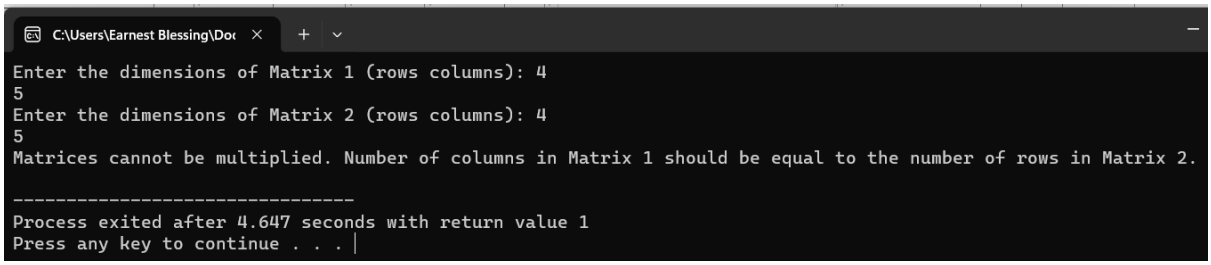
46.

```
#include <iostream>

// Function to dynamically allocate memory for a matrix
int** allocateMatrix(int rows, int cols) {
    int** matrix = new int*[rows];
    for (int i = 0; i < rows; ++i) {
        matrix[i] = new int[cols];
    }
    return matrix;
}

// Function to deallocate memory for a matrix
void deallocateMatrix(int** matrix, int rows) {
    for (int i = 0; i < rows; ++i) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

// Function to multiply two matrices
void multiplyMatrices(int** matrix1, int** matrix2, int** result, int rows1, int cols1, int rows2, int
cols2) {
    for (int i = 0; i < rows1; ++i) {
        for (int j = 0; j < cols2; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < cols1; ++k) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
```

```
    }
}

// Function to display the elements in a matrix
void displayMatrix(int** matrix, int rows, int cols) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
}

int main() {
    int rows1, cols1, rows2, cols2;

    // Get the dimensions of the matrices from the user
    std::cout << "Enter the dimensions of Matrix 1 (rows columns): ";
    std::cin >> rows1 >> cols1;

    std::cout << "Enter the dimensions of Matrix 2 (rows columns): ";
    std::cin >> rows2 >> cols2;

    // Check if matrices can be multiplied
    if (cols1 != rows2) {
        std::cerr << "Matrices cannot be multiplied. Number of columns in Matrix 1 should be equal to
the number of rows in Matrix 2." << std::endl;
        return 1;
    }

    // Allocate memory for matrices
```

```
int** matrix1 = allocateMatrix(rows1, cols1);
int** matrix2 = allocateMatrix(rows2, cols2);
int** resultMatrix = allocateMatrix(rows1, cols2);

// Input values for the matrices
std::cout << "Enter values for Matrix 1:" << std::endl;
for (int i = 0; i < rows1; ++i) {
    for (int j = 0; j < cols1; ++j) {
        std::cin >> matrix1[i][j];
    }
}

std::cout << "Enter values for Matrix 2:" << std::endl;
for (int i = 0; i < rows2; ++i) {
    for (int j = 0; j < cols2; ++j) {
        std::cin >> matrix2[i][j];
    }
}

// Multiply matrices
multiplyMatrices(matrix1, matrix2, resultMatrix, rows1, cols1, rows2, cols2);

// Display matrices and the result
std::cout << "\nMatrix 1:" << std::endl;
displayMatrix(matrix1, rows1, cols1);

std::cout << "\nMatrix 2:" << std::endl;
displayMatrix(matrix2, rows2, cols2);

std::cout << "\nProduct of Matrices:" << std::endl;
displayMatrix(resultMatrix, rows1, cols2);
```

```
    // Deallocate memory for matrices
    deallocateMatrix(matrix1, rows1);
    deallocateMatrix(matrix2, rows2);
    deallocateMatrix(resultMatrix, rows1);

    return 0;
}
```



47.

```
#include <iostream>

// Function to find the sum of diagonals of a square matrix
int sumOfDiagonals(int** matrix, int size) {
    int sum = 0;

    // Sum of main diagonal elements
    for (int i = 0; i < size; ++i) {
        sum += matrix[i][i];
    }

    // Sum of secondary diagonal elements
    for (int i = 0; i < size; ++i) {
        sum += matrix[i][size - i - 1];
    }
```

```
    return sum;
}

// Function to dynamically allocate memory for a square matrix
int** allocateSquareMatrix(int size) {
    int** matrix = new int*[size];
    for (int i = 0; i < size; ++i) {
        matrix[i] = new int[size];
    }
    return matrix;
}

// Function to deallocate memory for a square matrix
void deallocateSquareMatrix(int** matrix, int size) {
    for (int i = 0; i < size; ++i) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

// Function to display the elements in a square matrix
void displaySquareMatrix(int** matrix, int size) {
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
}
```

```
int main() {
    int size;

    // Get the size of the square matrix from the user
    std::cout << "Enter the size of the square matrix: ";
    std::cin >> size;

    // Allocate memory for the square matrix
    int** matrix = allocateSquareMatrix(size);

    // Input values for the matrix
    std::cout << "Enter values for the square matrix:" << std::endl;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            std::cin >> matrix[i][j];
        }
    }

    // Display the matrix
    std::cout << "\nSquare Matrix:" << std::endl;
    displaySquareMatrix(matrix, size);

    // Find and display the sum of diagonals
    int sum = sumOfDiagonals(matrix, size);
    std::cout << "\nSum of Diagonals: " << sum << std::endl;

    // Deallocate memory for the matrix
    deallocateSquareMatrix(matrix, size);

    return 0;
}
```

C:\Users\Earnest Blessing\Doc

```
Enter the size of the square matrix: 1
Enter values for the square matrix:
1

Square Matrix:
1

Sum of Diagonals: 2

--------------------------------
Process exited after 4.462 seconds with return value 0
Press any key to continue . . .
```