Contents

[**Hello World Program** 3](#_Toc164162292)

[**Check if a number is positive, negative, or zero** 3](#_Toc164162293)

[**Even or Odd Number Checker** 4](#_Toc164162294)

[**Swapping Two Numbers** 5](#_Toc164162295)

[**Swap two numbers without using a temporary variable** 6](#_Toc164162296)

[**Prime Number Checker** 6](#_Toc164162297)

[**Check if a Number is a Perfect Square** 7](#_Toc164162298)

[**Factorial Calculator** 8](#_Toc164162299)

[**Palindrome Checker for String** 9](#_Toc164162300)

[**Palindrome Checker for Number** 10](#_Toc164162301)

[**Calculate Power using Recursion** 11](#_Toc164162302)

[**Find the Maximum Number in an Array** 12](#_Toc164162303)

[**Reverse a Number** 12](#_Toc164162304)

[**String Reversal** 13](#_Toc164162305)

[**Count the number of vowels and consonants in a string** 14](#_Toc164162306)

[**Convert a String to Lowercase** 15](#_Toc164162307)

[**Convert a String to Uppercase** 15](#_Toc164162308)

[**Counts number of Words in a Sentence** 16](#_Toc164162309)

[**Remove all white spaces from a string** 16](#_Toc164162310)

[**Find the maximum occurring character in a string** 17](#_Toc164162311)

[**Convert a number to words** 18](#_Toc164162312)

[**Print ASCII Value of a Character** 21](#_Toc164162313)

[**Convert a decimal to any base (binary, octal, hexadecimal)** 21](#_Toc164162314)

[**Convert Decimal to Binary** 22](#_Toc164162315)

[**Convert Binary to Decimal** 23](#_Toc164162316)

[**Convert Decimal to Hexadecimal** 23](#_Toc164162317)

[**Convert Hexadecimal to Decimal** 24](#_Toc164162318)

[**Generate Random Numbers** 25](#_Toc164162319)

[**Random Password Generator** 25](#_Toc164162320)

[**Check Leap Year** 27](#_Toc164162321)

[**Find LCM (Least Common Multiple) of Two Numbers** 28](#_Toc164162322)

[**Find HCF (Highest Common Factor) of Two Numbers** 29](#_Toc164162323)

[**Calculator Program (Addition, Subtraction, Multiplication, Division)** 30](#_Toc164162324)

[**Simple Interest Calculator** 31](#_Toc164162325)

[**Calculate Compound Interest** 32](#_Toc164162326)

[**Area and Perimeter Calculator for Shapes (Circle, Square, Triangle)** 33](#_Toc164162327)

[**Temperature Converter (Celsius to Fahrenheit and vice versa)** 35](#_Toc164162328)

[**Currency Convertor (USD TO INR)** 36](#_Toc164162329)

[**Simple Alarm Clock** 39](#_Toc164162330)

[**Simple Stopwatch** 41](#_Toc164162331)

[**Simple Log in Application** 44](#_Toc164162332)

[**Simple Console-based Notes Application** 47](#_Toc164162333)

[**Simple To-Do List Application** 51](#_Toc164162334)

[**ATM Simulator** 56](#_Toc164162335)

[**Student Grade Calculator** 58](#_Toc164162336)

[**GPA Calculator for BCA Student** 59](#_Toc164162337)

[**File Manipulation (Read, Write, Append to File)** 63](#_Toc164162338)

[**Simple Chat Application** 65](#_Toc164162339)

[**Encryption and Decryption Program (e.g., Caesar Cipher)** 71](#_Toc164162340)

[**Simple Web Scraper (retrieve data from a website)** 73](#_Toc164162341)

[**Console Application for Web Scraping HTML Content** 73](#_Toc164162342)

[**Regular Expression Example (Email Validation)** 75](#_Toc164162343)

[**JSON Serialization and Deserialization** 76](#_Toc164162344)

[**Guess the Number Game** 77](#_Toc164162345)

[**Rock, Paper, Scissors Game** 79](#_Toc164162346)

[**Tic-Tac-Toe Game** 80](#_Toc164162347)

[**Hangman Game** 84](#_Toc164162348)

[**Console Based MCQs Question Paper** 87](#_Toc164162349)

[**Console Based Mini Address Book** 94](#_Toc164162350)

# **Hello World Program**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Hello, World!");

}

}

# **Check if a number is positive, negative, or zero**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a number: ");

int number = Convert.ToInt32(Console.ReadLine());

if (number > 0)

{

Console.WriteLine("Positive number");

}

else if (number < 0)

{

Console.WriteLine("Negative number");

}

else

{

Console.WriteLine("Zero");

}

}

}

# **Even or Odd Number Checker**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a number:");

int number = Convert.ToInt32(Console.ReadLine());

if (number % 2 == 0)

{  
 Console.WriteLine("The number is even.");

}

Else

{

Console.WriteLine("The number is odd.");

}

}

}

# **Swapping Two Numbers**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter first number: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter second number: ");

int num2 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Before swapping: num1 = " + num1 + " ,num2 = " + num2);

int temp = num1;

num1 = num2;

num2 = temp;

Console.WriteLine("After swapping: num1 = " + num1 + " ,num2 = " + num2);

}

}

# **Swap two numbers without using a temporary variable**

using System;

class Program

{

static void Main(string[] args)

{

int a = 5, b = 10;

Console.WriteLine("Before swapping: a = " + a + ", b = " + b);

a = a + b;

b = a - b;

a = a - b;

Console.WriteLine("After swapping: a = " + a + ", b = " + b);

}

}

# **Prime Number Checker**

using System;

class PrimeNumberFinder

{

static void Main()

{

Console.WriteLine("Enter a number to check if it's prime:");

int number = Convert.ToInt32(Console.ReadLine());

bool isPrime = true;

if (number <= 1)

isPrime = false;

for (int i = 2; i < number; i++)

{

if (number % i == 0)

{

isPrime = false;

break;

}

}

Console.WriteLine(isPrime ? $"{number} is a prime number." : $"{number} is not a prime number.");

}

}

# **Check if a Number is a Perfect Square**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a number: ");

int num = Convert.ToInt32(Console.ReadLine());

if (IsPerfectSquare(num))

Console.WriteLine("{0} is a perfect square", num);

else

Console.WriteLine("{0} is not a perfect square", num);

}

static bool IsPerfectSquare(int num) {

int sqrt = (int)Math.Sqrt(num);

return sqrt \* sqrt == num;

}

}

# **Factorial Calculator**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a number:");

int number = Convert.ToInt32(Console.ReadLine());

int factorial = 1;

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

{

factorial \*= i;

}

Console.WriteLine("Factorial of " + number + " is: " + factorial);

}

}

# **Palindrome Checker for String**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a number: ");

int num = Convert.ToInt32(Console.ReadLine());

int originalNum = num;

int reversedNum = 0;

while (num > 0) {

int remainder = num % 10;

reversedNum = reversedNum \* 10 + remainder;

num /= 10;

}

if (originalNum == reversedNum)

Console.WriteLine("{0} is a palindrome number", originalNum);

else

Console.WriteLine("{0} is not a palindrome number", originalNum);

}

}

# **Palindrome Checker for Number**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a number: ");

int num = Convert.ToInt32(Console.ReadLine());

int originalNum = num;

int reversedNum = 0;

while (num > 0) {

int digit = num % 10;

reversedNum = reversedNum \* 10 + digit;

num /= 10;

}

if (originalNum == reversedNum)

Console.WriteLine("{0} is a palindrome number", originalNum);

else

Console.WriteLine("{0} is not a palindrome number", originalNum);

}

}

# **Calculate Power using Recursion**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter the base: ");

double num = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter the exponent: ");

int exp = Convert.ToInt32(Console.ReadLine());

double result = Power(num, exp);

Console.WriteLine("{0} raised to the power of {1} is {2}", num, exp, result);

}

static double Power(double num, int exp) {

if (exp == 0)

return 1;

if (exp > 0)

return num \* Power(num, exp - 1);

else

return 1 / (num \* Power(num, -exp - 1));

}

}

# **Find the Maximum Number in an Array**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter the number of elements: ");

int n = Convert.ToInt32(Console.ReadLine());

int[] arr = new int[n];

Console.WriteLine("Enter the elements:");

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

arr[i] = Convert.ToInt32(Console.ReadLine());

}

int max = arr[0];

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

if (arr[i] > max)

max = arr[i];

}

Console.WriteLine("Maximum number in the array is: {0}", max);

}

}

# **Reverse a Number**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a number: ");

int num = Convert.ToInt32(Console.ReadLine());

int reversedNum = 0;

while (num > 0) {

int remainder = num % 10;

reversedNum = reversedNum \* 10 + remainder;

num /= 10;

}

# **String Reversal**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a string:");

string input = Console.ReadLine();

char[] charArray = input.ToCharArray();

Array.Reverse(charArray);

string reversedString = new string(charArray);

Console.WriteLine("Reversed string: " + reversedString);

}

}

# **Count the number of vowels and consonants in a string**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string str = Console.ReadLine().ToLower();

int vowels = 0, consonants = 0;

foreach (char ch in str)

{

if (ch >= 'a' && ch <= 'z')

{

if ("aeiou".Contains(ch))

vowels++;

else

consonants++;

}

}

Console.WriteLine("Vowels: " + vowels);

Console.WriteLine("Consonants: " + consonants);

}

}

# **Convert a String to Lowercase**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string str = Console.ReadLine();

string lowerCaseStr = str.ToLower();

Console.WriteLine("Lowercase string: " + lowerCaseStr);

}

}

# **Convert a String to Uppercase**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string str = Console.ReadLine();

string upperCaseStr = str.ToUpper();

Console.WriteLine("Uppercase string: " + upperCaseStr);

}

}

# **Counts number of Words in a Sentence**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a sentence: ");

string sentence = Console.ReadLine();

int wordCount = CountWords(sentence);

Console.WriteLine("Number of words in the sentence: {0}", wordCount);

}

static int CountWords(string sentence) {

string[] words = sentence.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

return words.Length;

}

}

# **Remove all white spaces from a string**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string str = Console.ReadLine();

string stringWithoutSpaces = str.Replace(" ", "");

Console.WriteLine("String without spaces: " + stringWithoutSpaces);

}

}

# **Find the maximum occurring character in a string**

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string str = Console.ReadLine();

Dictionary<char, int> charCount = new Dictionary<char, int>();

foreach (char ch in str)

{

if (charCount.ContainsKey(ch))

charCount[ch]++;

else

charCount[ch] = 1;

}

char maxChar = charCount.OrderByDescending(x => x.Value).First().Key;

Console.WriteLine("Maximum occurring character: " + maxChar);

}

}

# **Convert a number to words**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a number: ");

int number = int.Parse(Console.ReadLine());

Console.WriteLine(NumberToWords(number));

}

static string NumberToWords(int number)

{

if (number == 0)

return "Zero";

string words = "";

if ((number / 1000000) > 0)

{

words += NumberToWords(number / 1000000) + " Million ";

number %= 1000000;

}

if ((number / 1000) > 0)

{

words += NumberToWords(number / 1000) + " Thousand ";

number %= 1000;

}

if ((number / 100) > 0)

{

words += NumberToWords(number / 100) + " Hundred ";

number %= 100;

}

if (number > 0)

{

if (words != "")

words += "and ";

string[] units = { "", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine" };

string[] teens = { "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen" };

string[] tens = { "", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety" };

if (number < 10)

words += units[number];

else if (number < 20)

words += teens[number - 10];

else

{

words += tens[number / 10];

if ((number % 10) > 0)

words += "-" + units[number % 10];

}

}

return words;

}

}

# **Print ASCII Value of a Character**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a character: ");

char ch = Convert.ToChar(Console.ReadLine());

int asciiValue = (int)ch;

Console.WriteLine("ASCII value of {0} is {1}", ch, asciiValue);

}

}

# **Convert a decimal to any base (binary, octal, hexadecimal)**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a decimal number: ");

int decimalNumber = int.Parse(Console.ReadLine());

Console.WriteLine("Binary: " + Convert.ToString(decimalNumber, 2));

Console.WriteLine("Octal: " + Convert.ToString(decimalNumber, 8));

Console.WriteLine("Hexadecimal: " + Convert.ToString(decimalNumber, 16));

}

}

# **Convert Decimal to Binary**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a decimal number: ");

int decimalNum = Convert.ToInt32(Console.ReadLine());

string binaryNum = DecimalToBinary(decimalNum);

Console.WriteLine("Binary representation: {0}", binaryNum);

}

static string DecimalToBinary(int decimalNum) {

return Convert.ToString(decimalNum, 2);

}

}

# **Convert Binary to Decimal**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a binary number: ");

string binaryNum = Console.ReadLine();

int decimalNum = BinaryToDecimal(binaryNum);

Console.WriteLine("Decimal representation: {0}", decimalNum);

}

static int BinaryToDecimal(string binaryNum) {

return Convert.ToInt32(binaryNum, 2);

}

}

# **Convert Decimal to Hexadecimal**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a decimal number: ");

int decimalNum = Convert.ToInt32(Console.ReadLine());

string hexadecimalNum = DecimalToHexadecimal(decimalNum);

Console.WriteLine("Hexadecimal representation: {0}", hexadecimalNum);

}

static string DecimalToHexadecimal(int decimalNum) {

return decimalNum.ToString("X");

}

}

# **Convert Hexadecimal to Decimal**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a hexadecimal number: ");

string hexadecimalNum = Console.ReadLine();

int decimalNum = HexadecimalToDecimal(hexadecimalNum);

Console.WriteLine("Decimal representation: {0}", decimalNum);

}

static int HexadecimalToDecimal(string hexadecimalNum) {

return Convert.ToInt32(hexadecimalNum, 16);

}

}

# **Generate Random Numbers**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter the number of random numbers to generate: ");

int count = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter the minimum value: ");

int min = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter the maximum value: ");

int max = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Random Numbers:");

Random rand = new Random();

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

Console.WriteLine(rand.Next(min, max + 1));

}

}

}

# **Random Password Generator**

using System;

using System.Text;

class PasswordGenerator

{

private const string AllCharacters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()-\_";

public string Generate(int length)

{

StringBuilder password = new StringBuilder();

Random random = new Random();

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

{

int index = random.Next(0, AllCharacters.Length);

password.Append(AllCharacters[index]);

}

return password.ToString();

}

}

class Program

{

static void Main(string[] args)

{

PasswordGenerator generator = new PasswordGenerator();

Console.Write("Enter the length of the password: ");

int length = Convert.ToInt32(Console.ReadLine());

string password = generator.Generate(length);

Console.WriteLine("Generated Password: " + password);

}

}

# **Check Leap Year**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter a year: ");

int year = Convert.ToInt32(Console.ReadLine());

if (IsLeapYear(year))

Console.WriteLine("{0} is a leap year", year);

else

Console.WriteLine("{0} is not a leap year", year);

}

static bool IsLeapYear(int year) {

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

return true;

else

return false;

}

}

# **Find LCM (Least Common Multiple) of Two Numbers**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter first number: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter second number: ");

int num2 = Convert.ToInt32(Console.ReadLine());

int lcm = FindLCM(num1, num2);

Console.WriteLine("LCM of {0} and {1} is {2}", num1, num2, lcm);

}

static int FindLCM(int num1, int num2) {

int max = Math.Max(num1, num2);

int lcm = max;

while (true) {

if (lcm % num1 == 0 && lcm % num2 == 0)

break;

lcm += max;

}

return lcm;

}

}

# **Find HCF (Highest Common Factor) of Two Numbers**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter first number: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter second number: ");

int num2 = Convert.ToInt32(Console.ReadLine());

int hcf = FindHCF(num1, num2);

Console.WriteLine("HCF of {0} and {1} is {2}", num1, num2, hcf);

}

static int FindHCF(int num1, int num2) {

while (num2 != 0) {

int temp = num2;

num2 = num1 % num2;

num1 = temp;

}

return num1;

}

}

# **Calculator Program (Addition, Subtraction, Multiplication, Division)**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter two numbers:");

double num1 = Convert.ToDouble(Console.ReadLine());

double num2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Choose an operation (+, -, \*, /):");

char op = Convert.ToChar(Console.ReadLine());

double result = 0;

switch (op)

{

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

if (num2 != 0)

result = num1 / num2;

else

Console.WriteLine("Cannot divide by zero.");

break;

default:

Console.WriteLine("Invalid operation.");

break;

}

Console.WriteLine("Result: " + result);

}

}

# **Simple Interest Calculator**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter principal amount:");

double principal = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter rate of interest:");

double rate = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter time period (in years):");

double time = Convert.ToDouble(Console.ReadLine());

double simpleInterest = (principal \* rate \* time) / 100;

Console.WriteLine("Simple Interest: " + simpleInterest);

}

}

# **Calculate Compound Interest**

using System;

class Program {

static void Main(string[] args) {

Console.Write("Enter principal amount: ");

double principal = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter annual interest rate (in percentage): ");

double rate = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter time period in years: ");

double time = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter the number of times interest is compounded per year: ");

int n = Convert.ToInt32(Console.ReadLine());

double compoundInterest = CalculateCompoundInterest(principal, rate, time, n);

Console.WriteLine("Compound Interest: {0}", compoundInterest);

}

static double CalculateCompoundInterest(double principal, double rate, double time, int n) {

double compoundInterest = principal \* Math.Pow(1 + (rate / (100 \* n)), n \* time) - principal;

return compoundInterest;

}

}

# **Area and Perimeter Calculator for Shapes (Circle, Square, Triangle)**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Choose a shape (1. Circle, 2. Square, 3. Triangle):");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1: // Circle

Console.WriteLine("Enter radius:");

double radius = Convert.ToDouble(Console.ReadLine());

double circleArea = Math.PI \* radius \* radius;

double circlePerimeter = 2 \* Math.PI \* radius;

Console.WriteLine("Area of Circle: " + circleArea);

Console.WriteLine("Perimeter of Circle: " + circlePerimeter);

break;

case 2: // Square

Console.WriteLine("Enter side length:");

double side = Convert.ToDouble(Console.ReadLine());

double squareArea = side \* side;

double squarePerimeter = 4 \* side;

Console.WriteLine("Area of Square: " + squareArea);

Console.WriteLine("Perimeter of Square: " + squarePerimeter);

break;

case 3: // Triangle

Console.WriteLine("Enter base length:");

double baseLength = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter height:");

double height = Convert.ToDouble(Console.ReadLine());

double triangleArea = 0.5 \* baseLength \* height;

Console.WriteLine("Area of Triangle: " + triangleArea);

// Perimeter calculation depends on the type of triangle

// For simplicity, assuming it's an equilateral triangle

double trianglePerimeter = 3 \* baseLength;

Console.WriteLine("Perimeter of Triangle: " + trianglePerimeter);

break;

default:

Console.WriteLine("Invalid choice.");

break;

}

}

}

# **Temperature Converter (Celsius to Fahrenheit and vice versa)**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Choose conversion type (1. Celsius to Fahrenheit, 2. Fahrenheit to Celsius):");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1: // Celsius to Fahrenheit

Console.WriteLine("Enter temperature in Celsius:");

double celsius = Convert.ToDouble(Console.ReadLine());

double fahrenheit = (celsius \* 9 / 5) + 32;

Console.WriteLine("Temperature in Fahrenheit: " + fahrenheit);

break;

case 2: // Fahrenheit to Celsius

Console.WriteLine("Enter temperature in Fahrenheit:");

double f = Convert.ToDouble(Console.ReadLine());

double c = (f - 32) \* 5 / 9;

Console.WriteLine("Temperature in Celsius: " + c);

break;

default:

Console.WriteLine("Invalid choice.");

break;

}

}

}

# **Currency Convertor (USD TO INR)**

using System;

class CurrencyConverter

{

private const double UsdToEuroRate = 0.83;

private const double UsdToGbpRate = 0.72;

private const double UsdToInrRate = 74.39; // Predefined exchange rate for USD to INR

public double ConvertUsdToEuro(double usdAmount)

{

return usdAmount \* UsdToEuroRate;

}

public double ConvertUsdToGbp(double usdAmount)

{

return usdAmount \* UsdToGbpRate;

}

public double ConvertUsdToInr(double usdAmount)

{

return usdAmount \* UsdToInrRate;

}

public double ConvertInrToUsd(double inrAmount)

{

return inrAmount / UsdToInrRate;

}

// Add more conversion methods here as needed

}

class Program

{

static void Main(string[] args)

{

CurrencyConverter converter = new CurrencyConverter();

Console.WriteLine("Select the conversion direction:");

Console.WriteLine("1. USD to INR");

Console.WriteLine("2. INR to USD");

Console.Write("Enter your choice: ");

int choice = Convert.ToInt32(Console.ReadLine());

double amount;

switch (choice)

{

case 1:

Console.Write("Enter the amount in USD: ");

amount = Convert.ToDouble(Console.ReadLine());

double inrAmount = converter.ConvertUsdToInr(amount);

Console.WriteLine($"Equivalent amount in INR: {inrAmount} INR");

break;

case 2:

Console.Write("Enter the amount in INR: ");

amount = Convert.ToDouble(Console.ReadLine());

double usdAmount = converter.ConvertInrToUsd(amount);

Console.WriteLine($"Equivalent amount in USD: {usdAmount} USD");

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

}

# **Simple Alarm Clock**

using System;

using System.Threading;

class AlarmClock

{

public void SetAlarm(int hours, int minutes)

{

DateTime alarmTime = DateTime.Today.AddHours(hours).AddMinutes(minutes);

TimeSpan timeToWait = alarmTime - DateTime.Now;

if (timeToWait.TotalMilliseconds < 0)

{

Console.WriteLine("Invalid time. Please set the alarm for a future time.");

return;

}

Console.WriteLine($"Alarm set for {alarmTime.ToString("hh:mm tt")}. Waiting for the alarm...");

Thread.Sleep((int)timeToWait.TotalMilliseconds);

Console.WriteLine("ALARM! Time to wake up!");

}

}

class Program

{

static void Main(string[] args)

{

AlarmClock alarm = new AlarmClock();

Console.Write("Enter the hour for the alarm: ");

int hour = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter the minute for the alarm: ");

int minute = Convert.ToInt32(Console.ReadLine());

alarm.SetAlarm(hour, minute);

}

}

# **Simple Stopwatch**

using System;

using System.Threading;

class Stopwatch

{

private DateTime startTime;

private bool isRunning;

public void Start()

{

if (!isRunning)

{

startTime = DateTime.Now;

isRunning = true;

Console.WriteLine("Stopwatch started.");

}

else

{

Console.WriteLine("Stopwatch is already running.");

}

}

public void Stop()

{

if (isRunning)

{

TimeSpan elapsedTime = DateTime.Now - startTime;

isRunning = false;

Console.WriteLine("Stopwatch stopped. Elapsed time: " + elapsedTime);

}

else

{

Console.WriteLine("Stopwatch is not running.");

}

}

public void Reset()

{

startTime = DateTime.Now;

Console.WriteLine("Stopwatch reset.");

}

}

class Program

{

static void Main(string[] args)

{

Stopwatch stopwatch = new Stopwatch();

while (true)

{

Console.WriteLine("1. Start");

Console.WriteLine("2. Stop");

Console.WriteLine("3. Reset");

Console.WriteLine("4. Exit");

Console.Write("Enter your choice: ");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

stopwatch.Start();

break;

case 2:

stopwatch.Stop();

break;

case 3:

stopwatch.Reset();

break;

case 4:

Environment.Exit(0);

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

Console.WriteLine();

}

}

}

# **Simple Log in Application**

using System;

using System.Collections.Generic;

class ChatApplication

{

private Dictionary<string, string> users = new Dictionary<string, string>();

public void Start()

{

Console.WriteLine("Welcome to the Simple Chat Application!");

while (true)

{

Console.WriteLine("1. Log in");

Console.WriteLine("2. Register");

Console.WriteLine("3. Exit");

Console.Write("Enter your choice: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

LogIn();

break;

case "2":

Register();

break;

case "3":

Console.WriteLine("Exiting...");

return;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

}

private void LogIn()

{

Console.Write("Enter your username: ");

string username = Console.ReadLine();

Console.Write("Enter your password: ");

string password = Console.ReadLine();

if (users.ContainsKey(username) && users[username] == password)

{

Console.WriteLine("Logged in successfully!");

// Implement chat functionality here

}

else

{

Console.WriteLine("Incorrect username or password. Please try again.");

}

}

private void Register()

{

Console.Write("Enter a new username: ");

string username = Console.ReadLine();

if (users.ContainsKey(username))

{

Console.WriteLine("Username already exists. Please choose a different username.");

return;

}

Console.Write("Enter a password: ");

string password = Console.ReadLine();

users.Add(username, password);

Console.WriteLine("Registration successful!");

}

}

class Program

{

static void Main(string[] args)

{

ChatApplication chatApp = new ChatApplication();

chatApp.Start();

}

}

# **Simple Console-based Notes Application**

using System;

using System.Collections.Generic;

using System.IO;

class Program

{

static string notesFilePath = "notes.txt";

static List<string> notes = new List<string>();

static void Main(string[] args)

{

LoadNotesFromFile();

while (true)

{

Console.WriteLine("\nNotes Menu:");

Console.WriteLine("1. Add Note");

Console.WriteLine("2. View Notes");

Console.WriteLine("3. Delete Note");

Console.WriteLine("4. Exit");

Console.Write("Enter your choice: ");

if (int.TryParse(Console.ReadLine(), out int choice))

{

switch (choice)

{

case 1:

AddNote();

break;

case 2:

ViewNotes();

break;

case 3:

DeleteNote();

break;

case 4:

SaveNotesToFile();

Console.WriteLine("Exiting...");

Environment.Exit(0);

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

else

{

Console.WriteLine("Invalid choice. Please enter a number.");

}

}

}

static void AddNote()

{

Console.WriteLine("Enter your note (press Enter on an empty line to finish):");

string input = Console.ReadLine();

while (!string.IsNullOrWhiteSpace(input))

{

notes.Add(input);

input = Console.ReadLine();

}

Console.WriteLine("Note added successfully!");

}

static void ViewNotes()

{

if (notes.Count > 0)

{

Console.WriteLine("\nYour Notes:");

for (int i = 0; i < notes.Count; i++)

{

Console.WriteLine($"{i + 1}. {notes[i]}");

}

}

else

{

Console.WriteLine("\nNo notes available.");

}

}

static void DeleteNote()

{

ViewNotes();

Console.Write("Enter the number of the note you want to delete: ");

if (int.TryParse(Console.ReadLine(), out int noteNumber) && noteNumber >= 1 && noteNumber <= notes.Count)

{

notes.RemoveAt(noteNumber - 1);

Console.WriteLine("Note deleted successfully!");

}

else

{

Console.WriteLine("Invalid note number.");

}

}

static void LoadNotesFromFile()

{

if (File.Exists(notesFilePath))

{

notes.AddRange(File.ReadAllLines(notesFilePath));

}

}

static void SaveNotesToFile()

{

File.WriteAllLines(notesFilePath, notes);

}

}

# **Simple To-Do List Application**

using System;

using System.Collections.Generic;

using System.IO;

class Program

{

static string filePath = "todo.txt";

static void Main(string[] args)

{

List<string> todoList = LoadTasksFromFile();

while (true)

{

Console.WriteLine("To-Do List Menu:");

Console.WriteLine("1. Add Task");

Console.WriteLine("2. View Tasks");

Console.WriteLine("3. Mark Task as Completed");

Console.WriteLine("4. Exit");

Console.Write("Enter your choice: ");

int choice;

if (int.TryParse(Console.ReadLine(), out choice))

{

switch (choice)

{

case 1:

Console.Write("Enter task: ");

string task = Console.ReadLine();

todoList.Add(task);

SaveTasksToFile(todoList);

Console.WriteLine("Task added successfully!");

break;

case 2:

ShowTasks(todoList);

break;

case 3:

MarkTaskAsCompleted(todoList);

break;

case 4:

Console.WriteLine("Exiting...");

Environment.Exit(0);

break;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

else

{

Console.WriteLine("Invalid choice. Please enter a number.");

}

Console.WriteLine();

}

}

static void ShowTasks(List<string> todoList)

{

if (todoList.Count > 0)

{

Console.WriteLine("Tasks:");

for (int i = 0; i < todoList.Count; i++)

{

Console.WriteLine($"{i + 1}. {todoList[i]}");

}

}

else

{

Console.WriteLine("No tasks in the list.");

}

}

static void MarkTaskAsCompleted(List<string> todoList)

{

if (todoList.Count > 0)

{

ShowTasks(todoList);

Console.Write("Enter task number to mark as completed: ");

int taskNumber;

if (int.TryParse(Console.ReadLine(), out taskNumber) && taskNumber >= 1 && taskNumber <= todoList.Count)

{

todoList.RemoveAt(taskNumber - 1);

SaveTasksToFile(todoList);

Console.WriteLine("Task marked as completed!");

}

else

{

Console.WriteLine("Invalid task number.");

}

}

else

{

Console.WriteLine("No tasks in the list.");

}

}

static List<string> LoadTasksFromFile()

{

List<string> tasks = new List<string>();

if (File.Exists(filePath))

{

tasks.AddRange(File.ReadAllLines(filePath));

}

return tasks;

}

static void SaveTasksToFile(List<string> todoList)

{

File.WriteAllLines(filePath, todoList);

}

}

# **ATM Simulator**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Welcome to the ATM Simulator!");

int balance = 10000;

while (true)

{

Console.WriteLine("\nSelect an option:");

Console.WriteLine("1. Check Balance");

Console.WriteLine("2. Deposit");

Console.WriteLine("3. Withdraw");

Console.WriteLine("4. Exit");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

Console.WriteLine($"Your balance is: {balance}");

break;

case 2:

Console.WriteLine("Enter deposit amount:");

int depositAmount = Convert.ToInt32(Console.ReadLine());

balance += depositAmount;

Console.WriteLine($"Deposit of {depositAmount} successful. Your new balance is: {balance}");

break;

case 3:

Console.WriteLine("Enter withdrawal amount:");

int withdrawalAmount = Convert.ToInt32(Console.ReadLine());

if (withdrawalAmount <= balance)

{

balance -= withdrawalAmount;

Console.WriteLine($"Withdrawal of {withdrawalAmount} successful. Your new balance is: {balance}");

}

else

{

Console.WriteLine("Insufficient funds.");

}

break;

case 4:

Console.WriteLine("Thank you for using the ATM Simulator. Goodbye!");

return;

default:

Console.WriteLine("Invalid option. Please try again.");

break;

}

}

}

}

# **Student Grade Calculator**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter the number of subjects:");

int numSubjects = Convert.ToInt32(Console.ReadLine());

double totalMarks = 0;

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

{

Console.WriteLine($"Enter marks for Subject {i}:");

double marks = Convert.ToDouble(Console.ReadLine());

totalMarks += marks;

}

double average = totalMarks / numSubjects;

Console.WriteLine($"Average marks: {average}");

if (average >= 90)

Console.WriteLine("Grade: A");

else if (average >= 80)

Console.WriteLine("Grade: B");

else if (average >= 70)

Console.WriteLine("Grade: C");

else if (average >= 60)

Console.WriteLine("Grade: D");

else

Console.WriteLine("Grade: F (Fail)");

}

}

# **GPA Calculator for BCA Student**

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

Console.WriteLine("Welcome to the IGNOU BCA GPA Calculator!");

Console.WriteLine("Enter your exam marks (out of 100) and assignment marks for each course. Type 'done' when finished.\n");

List<double> examMarks = new List<double>();

List<double> assignmentMarks = new List<double>();

List<int> credits = new List<int>();

string input;

// Prompt for exam marks, assignment marks, and credits until the user is done

do

{

Console.Write("Enter exam marks (out of 100) or 'done': ");

input = Console.ReadLine();

if (input.ToLower() != "done")

{

if (double.TryParse(input, out double examMark))

{

if (examMark >= 0 && examMark <= 100)

{

Console.Write("Enter assignment marks (out of 100): ");

if (double.TryParse(Console.ReadLine(), out double assignmentMark))

{

Console.Write("Enter credits for the course: ");

if (int.TryParse(Console.ReadLine(), out int credit) && credit > 0)

{

examMarks.Add(examMark);

assignmentMarks.Add(assignmentMark);

credits.Add(credit);

}

else

{

Console.WriteLine("Invalid credits. Please enter a positive integer.");

}

}

else

{

Console.WriteLine("Invalid assignment marks. Please enter a number.");

}

}

else

{

Console.WriteLine("Invalid exam marks. Please enter a number between 0 and 100.");

}

}

else

{

Console.WriteLine("Invalid input. Please enter a number or 'done'.");

}

}

} while (input.ToLower() != "done");

// Calculate GPA

double totalGradePoints = 0;

int totalCredits = 0;

for (int i = 0; i < examMarks.Count; i++)

{

// Calculate the total marks based on the weightage (75% exam + 25% assignment)

double totalMarks = (0.75 \* examMarks[i]) + (0.25 \* assignmentMarks[i]);

// Convert the total marks out of 100 to a GPA scale of 0 to 10

double gradePoint = (totalMarks / 100) \* 10;

// Add the weighted grade points based on the credits for each course

totalGradePoints += gradePoint \* credits[i];

// Add the total credits

totalCredits += credits[i];

}

// Calculate the GPA

double gpa = totalCredits > 0 ? totalGradePoints / totalCredits : 0;

Console.WriteLine($"Your GPA is: {gpa:F2}");

}

}

# **File Manipulation (Read, Write, Append to File)**

using System;

using System.IO;

class Program

{

static void Main()

{

string filePath = "example.txt";

// Write to file

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine("Hello, World!");

writer.WriteLine("This is a sample text file.");

}

// Read from file

Console.WriteLine("Contents of the file:");

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

// Append to file

using (StreamWriter writer = File.AppendText(filePath))

{

writer.WriteLine("This line is appended to the file.");

}

// Read updated file

Console.WriteLine("\nUpdated contents of the file:");

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

}

}

# **Simple Chat Application**

using System;

using System.Diagnostics;

using System.Web;

class Program

{

    static void Main()

    {

        Console.WriteLine("Welcome to the Enhanced Chat Application!");

        Console.WriteLine("Type 'exit' to end the chat.\n");

        while (true)

        {

            Console.Write("You: ");

            string message = Console.ReadLine();

            if (message.ToLower() == "exit")

                break;

            Console.WriteLine("Bot: " + GetResponse(message));

        }

    }

    static string GetResponse(string message)

    {

        // Common greetings

        if (message.ToLower().Contains("hello") || message.ToLower().Contains("hi"))

        {

            return "Hi there! How can I assist you today?";

        }

        // Farewell

        else if (message.ToLower().Contains("bye") || message.ToLower().Contains("goodbye"))

        {

            return "Goodbye! Have a great day!";

        }

        // Ask about the user's day

        else if (message.ToLower().Contains("how are you") || message.ToLower().Contains("how's it going"))

        {

            return "I'm just a bot, but thanks for asking! How can I make your day better?";

        }

        // Talk about hobbies

        else if (message.ToLower().Contains("hobby") || message.ToLower().Contains("interest"))

        {

            return "I like chatting with people! What about you? Any favorite hobbies?";

        }

        // Jokes

        else if (message.ToLower().Contains("joke") || message.ToLower().Contains("funny"))

        {

            return "Why don't scientists trust atoms? Because they make up everything!";

        }

        // Ask about the user's favorite movie

        else if (message.ToLower().Contains("movie") || message.ToLower().Contains("film"))

        {

            return "Movies are great! Do you have a favorite genre?";

        }

        // Ask about the user's favorite music

        else if (message.ToLower().Contains("music"))

        {

            return "Music is wonderful! What type of music do you enjoy listening to?";

        }

        // If the user asks about the weather

        else if (message.ToLower().Contains("weather"))

        {

            return "The weather is nice today! Would you like me to check the forecast for you?";

        }

        // If the user asks about the time

        else if (message.ToLower().Contains("time"))

        {

            return "It's currently " + DateTime.Now.ToString("hh:mm tt") + ". Today is " + DateTime.Now.ToString("dddd, MMMM dd, yyyy");

        }

        else if (message.ToLower().Contains("time"))

        {

            return "Today is " + DateTime.Now.ToString("dddd, MMMM dd, yyyy");

        }

        // Open YouTube

        else if (message.ToLower().Contains("youtube"))

        {

            OpenUrl("https://www.youtube.com/");

            return "Opening YouTube for you!";

        }

        // Open any website

        else if (message.ToLower().Contains("open "))

        {

            string url = message.ToLower().Replace("open ", "").Trim();

            if (!string.IsNullOrEmpty(url) && (url.StartsWith("https://") || url.StartsWith("https://")))

            {

                OpenUrl(url);

                return "Opening " + url + " for you!";

            }

            else

            {

                return "Sorry, I couldn't understand the website URL.";

            }

        }

        // Play music or videos on YouTube

        else if (message.ToLower().Contains("play "))

        {

            string searchQuery = message.ToLower().Replace("play ", "").Trim();

            if (!string.IsNullOrEmpty(searchQuery))

            {

                string searchUrl = "https://www.youtube.com/results?search\_query=" + HttpUtility.UrlEncode(searchQuery);

                OpenUrl(searchUrl);

                return "Playing music or videos related to '" + searchQuery + "' on YouTube!";

            }

            else

            {

                return "What would you like me to play?";

            }

        }

        // Weather forecast

        else if (message.ToLower().Contains("weather forecast"))

        {

            // You can integrate with a weather API here to fetch the forecast

            return "The weather forecast for your location is currently unavailable.";

        }

        // More advanced question and answer capabilities (can be implemented using NLP and ML techniques)

        else if (message.ToLower().Contains("question"))

        {

            // Implement more advanced question and answer logic here

            return "I'm still learning how to answer various questions. Ask me anything!";

        }

        // Handle unknown queries

        else

        {

            return "I'm not sure how to respond to that. Can you please provide more context?";

        }

    }

    static void OpenUrl(string url)

    {

        try

        {

            Process.Start(new ProcessStartInfo(url) { UseShellExecute = true });

        }

        catch (Exception ex)

        {

            Console.WriteLine("Error: " + ex.Message);

        }

    }

}

# **Encryption and Decryption Program (e.g., Caesar Cipher)**

using System;

class Program

{

static void Main()

{

string message = "Hello, World!";

int key = 3;

string encryptedMessage = Encrypt(message, key);

Console.WriteLine("Encrypted Message: " + encryptedMessage);

string decryptedMessage = Decrypt(encryptedMessage, key);

Console.WriteLine("Decrypted Message: " + decryptedMessage);

}

static string Encrypt(string message, int key)

{

string encryptedMessage = "";

foreach (char ch in message)

{

if (char.IsLetter(ch))

{

char shiftedChar = (char)(ch + key);

if ((char.IsLower(ch) && shiftedChar > 'z') || (char.IsUpper(ch) && shiftedChar > 'Z'))

shiftedChar = (char)(ch - (26 - key));

encryptedMessage += shiftedChar;

}

else

{

encryptedMessage += ch;

}

}

return encryptedMessage;

}

static string Decrypt(string encryptedMessage, int key)

{

return Encrypt(encryptedMessage, -key);

}

}

# **Simple Web Scraper (retrieve data from a website)**

using System;

using System.Net;

class Program

{

static void Main()

{

string url = "https://www.example.com";

string htmlContent = "";

using (WebClient client = new WebClient())

{

htmlContent = client.DownloadString(url);

}

Console.WriteLine(htmlContent);

}

}

# **Console Application for Web Scraping HTML Content**

using System;

using System.IO;

using System.Net;

using System.Threading.Tasks;

class Program

{

static async Task Main()

{

string url = "https://www.instagram.com";

string htmlContent = "";

try

{

using (HttpClient client = new HttpClient())

{

htmlContent = await client.GetStringAsync(url);

}

// Save HTML content to a file

string fileName = "instagram\_homepage.html";

File.WriteAllText(fileName, htmlContent);

Console.WriteLine($"HTML content saved to '{fileName}'");

}

catch (HttpRequestException ex)

{

Console.WriteLine($"Failed to download content: {ex.Message}");

}

catch (Exception ex)

{

Console.WriteLine($"An error occurred: {ex.Message}");

}

}

}

# **Regular Expression Example (Email Validation)**

using System;

using System.Text.RegularExpressions;

class Program

{

static void Main()

{

string email = "example@email.com";

if (IsValidEmail(email))

{

Console.WriteLine("Valid email address.");

}

else

{

Console.WriteLine("Invalid email address.");

}

}

static bool IsValidEmail(string email)

{

string pattern = @"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$";

return Regex.IsMatch(email, pattern);

}

}

# **JSON Serialization and Deserialization**

using System;

using System.IO;

using System.Text.Json;

class Program

{

static void Main()

{

// Serialization

Person person = new Person { Name = "John", Age = 30 };

string jsonString = JsonSerializer.Serialize(person);

Console.WriteLine("Serialized JSON:");

Console.WriteLine(jsonString);

// Deserialization

Person deserializedPerson = JsonSerializer.Deserialize<Person>(jsonString);

Console.WriteLine("\nDeserialized Object:");

Console.WriteLine($"Name: {deserializedPerson.Name}, Age: {deserializedPerson.Age}");

}

}

class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

# **Guess the Number Game**

using System;

class Program

{

static void Main()

{

Random random = new Random();

int targetNumber = random.Next(1, 101);

int attempts = 0;

Console.WriteLine("Guess the number between 1 and 100:");

while (true)

{

int guess = Convert.ToInt32(Console.ReadLine());

attempts++;

if (guess == targetNumber)

{

Console.WriteLine($"Congratulations! You guessed the number in {attempts} attempts.");

break;

}

else if (guess < targetNumber)

{

Console.WriteLine("Too low. Try again:");

}

else

{

Console.WriteLine("Too high. Try again:");

}

}

}

}

# **Rock, Paper, Scissors Game**

using System;

class Program

{

static void Main()

{

string[] options = { "rock", "paper", "scissors" };

Console.WriteLine("Enter your choice (rock, paper, or scissors):");

string userChoice = Console.ReadLine().ToLower();

Random random = new Random();

int index = random.Next(0, 3);

string computerChoice = options[index];

Console.WriteLine($"Computer's choice: {computerChoice}");

if (userChoice == computerChoice)

{

Console.WriteLine("It's a tie!");

}

else if ((userChoice == "rock" && computerChoice == "scissors") ||

(userChoice == "paper" && computerChoice == "rock") ||

(userChoice == "scissors" && computerChoice == "paper"))

{

Console.WriteLine("You win!");

}

else

{

Console.WriteLine("Computer wins!");

}

}

}

# **Tic-Tac-Toe Game**

using System;

class Program

{

static char[,] board = new char[3, 3];

static char currentPlayer = 'X';

static void Main()

{

InitializeBoard();

while (true)

{

DisplayBoard();

Console.WriteLine($"Player {currentPlayer}'s turn. Enter row and column (e.g., 1 1):");

string[] input = Console.ReadLine().Split();

int row = Convert.ToInt32(input[0]) - 1;

int col = Convert.ToInt32(input[1]) - 1;

if (IsValidMove(row, col))

{

MakeMove(row, col);

if (IsWinner() || IsBoardFull())

{

DisplayBoard();

if (IsWinner())

Console.WriteLine($"Player {currentPlayer} wins!");

else

Console.WriteLine("It's a tie!");

break;

}

TogglePlayer();

}

else

{

Console.WriteLine("Invalid move. Try again.");

}

}

}

static void InitializeBoard()

{

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

{

for (int j = 0; j < 3; j++)

{

board[i, j] = '-';

}

}

}

static void DisplayBoard()

{

Console.WriteLine(" 1 2 3");

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

{

Console.Write($"{i + 1} ");

for (int j = 0; j < 3; j++)

{

Console.Write($"{board[i, j]} ");

}

Console.WriteLine();

}

}

static bool IsValidMove(int row, int col)

{

return row >= 0 && row < 3 && col >= 0 && col < 3 && board[row, col] == '-';

}

static void MakeMove(int row, int col)

{

board[row, col] = currentPlayer;

}

static void TogglePlayer()

{

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

}

static bool IsWinner()

{

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

{

if (board[i, 0] != '-' && board[i, 0] == board[i, 1] && board[i, 1] == board[i, 2])

return true;

if (board[0, i] != '-' && board[0, i] == board[1, i] && board[1, i] == board[2, i])

return true;

}

if (board[0, 0] != '-' && board[0, 0] == board[1, 1] && board[1, 1] == board[2, 2])

return true;

if (board[0, 2] != '-' && board[0, 2] == board[1, 1] && board[1, 1] == board[2, 0])

return true;

return false;

}

static bool IsBoardFull()

{

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

{

for (int j = 0; j < 3; j++)

{

if (board[i, j] == '-')

return false;

}

}

return true;

}

}

# **Hangman Game**

using System;

class Program

{

static string[] words = { "apple", "banana", "cherry", "orange", "strawberry" };

static string wordToGuess;

static char[] guessedWord;

static int attemptsLeft = 6;

static void Main()

{

Random random = new Random();

wordToGuess = words[random.Next(words.Length)];

guessedWord = new char[wordToGuess.Length];

for (int i = 0; i < wordToGuess.Length; i++)

{

guessedWord[i] = '\_';

}

Console.WriteLine("Welcome to Hangman!");

Console.WriteLine("Guess the fruit:");

DisplayGuessedWord();

while (attemptsLeft > 0 && !IsWordGuessed())

{

Console.WriteLine($"Attempts left: {attemptsLeft}");

Console.WriteLine("Enter a letter:");

char guess = Convert.ToChar(Console.ReadLine());

if (IsGuessCorrect(guess))

{

Console.WriteLine("Correct!");

}

else

{

Console.WriteLine("Incorrect!");

attemptsLeft--;

}

DisplayGuessedWord();

}

if (IsWordGuessed())

Console.WriteLine("Congratulations! You guessed the word.");

else

Console.WriteLine($"You ran out of attempts. The word was '{wordToGuess}'.");

}

static void DisplayGuessedWord()

{

foreach (char c in guessedWord)

{

Console.Write($"{c} ");

}

Console.WriteLine();

}

static bool IsGuessCorrect(char guess)

{

bool correctGuess = false;

for (int i = 0; i < wordToGuess.Length; i++)

{

if (wordToGuess[i] == guess)

{

guessedWord[i] = guess;

correctGuess = true;

}

}

return correctGuess;

}

static bool IsWordGuessed()

{

return Array.IndexOf(guessedWord, '\_') == -1;

}

}

# **Console Based MCQs Question Paper**

using System;

using System.Collections.Generic;

using System.IO;

class Program

{

    static void Main(string[] args)

    {

        string[] questions = new string[]

        {

            "1. What does C# stand for?",

            "2. Which of the following is NOT a valid C# data type?",

            "3. What does the 'using' keyword do in C#?",

            "4. Which of the following is the correct way to declare a variable in C#?",

            "5. What is the purpose of the 'static' keyword in C#?",

            "6. What is the output of the following code snippet?\n   int x = 10;\n   Console.WriteLine(x++);",

            "7. Which of the following access modifiers makes a member accessible only within its own class?",

            "8. What is the role of the 'this' keyword in C#?",

            "9. What does the 'sealed' keyword signify in C#?",

            "10. What is the default access modifier for members of a class in C#?",

            "11. Which of the following is NOT a valid way to instantiate an object in C#?",

            "12. What is the purpose of the 'as' operator in C#?",

            "13. What does the 'params' keyword allow you to do in C#?",

            "14. What is the difference between 'throw' and 'throw ex' in C#?",

            "15. What is the purpose of the 'finally' block in a try-catch-finally statement in C#?",

            "16. Which of the following is NOT a valid type of loop in C#?",

            "17. What does the 'break' keyword do in a loop in C#?",

            "18. What is the purpose of the 'using' statement in C#?",

            "19. Which of the following is a feature of C# 8.0?",

            "20. What is the output of the following code snippet?\n    Console.WriteLine(\"Hello\" + 1 + 2);"

        };

        string[][] options = new string[][]

        {

            new string[] { "a. C Sharp", "b. C Sharpie", "c. C Hash", "d. C Number" },

            new string[] { "a. int", "b. string", "c. char[]", "d. real" },

            new string[] { "a. Declares a namespace", "b. Imports a namespace", "c. Allocates memory", "d. Defines a class" },

            new string[] { "a. int x;", "b. x = 10;", "c. int x = 10;", "d. x: int = 10;" },

            new string[] { "a. It makes a variable accessible from any instance of the class.", "b. It allows a method to be called without creating an instance of the class.", "c. It indicates that a method is an instance method.", "d. It prevents a class from being instantiated." },

            new string[] { "a. 10", "b. 11", "c. 9", "d. 0" },

            new string[] { "a. private", "b. protected", "c. internal", "d. public" },

            new string[] { "a. It refers to the current instance of the class.", "b. It creates a new instance of the class.", "c. It specifies that a method is static.", "d. It is used to call a base class constructor." },

            new string[] { "a. It prevents inheritance of a class.", "b. It indicates that a method cannot be overridden.", "c. It prevents modification of a method.", "d. It indicates that a method cannot be called." },

            new string[] { "a. private", "b. protected", "c. internal", "d. public" },

            new string[] { "a. MyClass obj = new MyClass();", "b. MyClass obj();", "c. MyClass obj = new();", "d. MyClass obj = CreateObject();" },

            new string[] { "a. It performs a type conversion.", "b. It specifies the access level of a class.", "c. It checks for null references.", "d. It is used to cast an object to a derived type." },

            new string[] { "a. It specifies a variable number of arguments for a method.", "b. It indicates that a method can be called with a variable number of arguments.", "c. It specifies a variable number of return values for a method.", "d. It allows optional parameters in a method." },

            new string[] { "a. They are equivalent.", "b. 'throw ex' preserves the original stack trace.", "c. 'throw' preserves the original stack trace.", "d. 'throw' rethrows the caught exception." },

            new string[] { "a. It is executed if an exception is thrown.", "b. It is executed regardless of whether an exception is thrown.", "c. It is executed before the try block.", "d. It is executed after the catch block." },

            new string[] { "a. Foreach", "b. While", "c. Until", "d. For" },

            new string[] { "a. Exits the loop immediately.", "b. Continues to the next iteration of the loop.", "c. Restarts the loop from the beginning.", "d. Skips the current iteration of the loop." },

            new string[] { "a. It ensures that an object is disposed of properly.", "b. It is used to include a namespace in a C# file.", "c. It is used to allocate memory dynamically.", "d. It is used to declare a variable." },

            new string[] { "a. Nullable reference types", "b. Asynchronous streams", "c. Default interface methods", "d. Pattern matching enhancements" },

            new string[] { "a. Hello12", "b. Hello3", "c. 12Hello", "d. Compilation error" }

        };

        char[] answers = new char[] { 'a', 'd', 'b', 'c', 'b', 'a', 'a', 'a', 'a', 'd', 'b', 'a', 'a', 'b', 'b', 'c', 'a', 'a', 'd', 'a' };

        List<char> userResponses = new List<char>();

        Console.WriteLine("Welcome to the C# Programming Quiz!");

        Console.WriteLine("Select the correct option for each question. (a, b, c, or d)");

        Console.WriteLine("Type 'quit' to end the quiz. Type 'reset' to start over.\n");

        for (int i = 0; i < questions.Length; i++)

        {

            Console.WriteLine(questions[i]);

            foreach (string option in options[i])

            {

                Console.WriteLine(option);

            }

            Console.Write("Your answer: ");

            string userInput = Console.ReadLine().ToLower();

            if (userInput == "quit")

            {

                Console.WriteLine("Quiz terminated.");

                break;

            }

            else if (userInput == "reset")

            {

                Console.WriteLine("Quiz reset.");

                userResponses.Clear();

                i = -1; // Reset to start from the first question again

            }

            else

            {

                char userAnswer = userInput[0];

                userResponses.Add(userAnswer);

                if (userAnswer == answers[i])

                {

                    Console.WriteLine("Correct!\n");

                }

                else

                {

                    Console.WriteLine("Incorrect. The correct answer is: " + answers[i] + "\n");

                }

            }

        }

        Console.WriteLine("Quiz complete!");

        // Save the answers to a file

        SaveAnswers(questions, userResponses);

        Console.WriteLine("Your answers have been saved.");

    }

    static void SaveAnswers(string[] questions, List<char> answers)

    {

        if (questions.Length == answers.Count)

        {

            using (StreamWriter writer = new StreamWriter("answers\_booklet.txt"))

            {

                writer.WriteLine("C# Programming Quiz - Answers Booklet\n");

                for (int i = 0; i < questions.Length; i++)

                {

                    writer.WriteLine($"{questions[i]} Answer: {answers[i]}");

                }

            }

            Console.WriteLine("Your answers have been saved.");

        }

        else

        {

            Console.WriteLine("Unable to save answers. Please complete the quiz first.");

        }

    }

}

# **Console Based Mini Address Book**

using System;

using System.Collections.Generic;

using System.IO;

class Contact

{

public string Name { get; set; }

public string Phone { get; set; }

public string Email { get; set; }

public override string ToString()

{

return $"Name: {Name}, Phone: {Phone}, Email: {Email}";

}

}

class Program

{

static string contactsFilePath = "contacts.txt";

static List<Contact> contacts = new List<Contact>();

static void Main(string[] args)

{

LoadContactsFromFile();

while (true)

{

Console.WriteLine("\nAddress Book Menu:");

Console.WriteLine("1. Add Contact");

Console.WriteLine("2. View Contacts");

Console.WriteLine("3. Search Contact");

Console.WriteLine("4. Delete Contact");

Console.WriteLine("5. Exit");

Console.Write("Enter your choice: ");

if (int.TryParse(Console.ReadLine(), out int choice))

{

switch (choice)

{

case 1:

AddContact();

break;

case 2:

ViewContacts();

break;

case 3:

SearchContact();

break;

case 4:

DeleteContact();

break;

case 5:

SaveContactsToFile();

Console.WriteLine("Exiting...");

return;

default:

Console.WriteLine("Invalid choice. Please try again.");

break;

}

}

else

{

Console.WriteLine("Invalid choice. Please enter a number.");

}

}

}

static void AddContact()

{

Console.WriteLine("Enter contact details:");

Contact contact = new Contact();

Console.Write("Name: ");

contact.Name = Console.ReadLine();

Console.Write("Phone: ");

contact.Phone = Console.ReadLine();

Console.Write("Email: ");

contact.Email = Console.ReadLine();

contacts.Add(contact);

Console.WriteLine("Contact added successfully!");

}

static void ViewContacts()

{

if (contacts.Count > 0)

{

Console.WriteLine("\nContacts:");

for (int i = 0; i < contacts.Count; i++)

{

Console.WriteLine($"{i + 1}. {contacts[i]}");

}

}

else

{

Console.WriteLine("\nNo contacts available.");

}

}

static void SearchContact()

{

Console.Write("Enter the name to search: ");

string searchName = Console.ReadLine();

List<Contact> searchResults = new List<Contact>();

foreach (var contact in contacts)

{

if (contact.Name.Contains(searchName))

{

searchResults.Add(contact);

}

}

if (searchResults.Count > 0)

{

Console.WriteLine("\nSearch Results:");

foreach (var contact in searchResults)

{

Console.WriteLine(contact);

}

}

else

{

Console.WriteLine("\nNo matching contacts found.");

}

}

static void DeleteContact()

{

ViewContacts();

Console.Write("Enter the index of the contact you want to delete: ");

if (int.TryParse(Console.ReadLine(), out int index) && index >= 1 && index <= contacts.Count)

{

contacts.RemoveAt(index - 1);

Console.WriteLine("Contact deleted successfully!");

}

else

{

Console.WriteLine("Invalid contact index.");

}

}

static void LoadContactsFromFile()

{

if (File.Exists(contactsFilePath))

{

string[] lines = File.ReadAllLines(contactsFilePath);

foreach (var line in lines)

{

string[] parts = line.Split(',');

if (parts.Length == 3)

{

Contact contact = new Contact { Name = parts[0], Phone = parts[1], Email = parts[2] };

contacts.Add(contact);

}

}

}

}

static void SaveContactsToFile()

{

List<string> lines = new List<string>();

foreach (var contact in contacts)

{

string line = $"{contact.Name},{contact.Phone},{contact.Email}";

lines.Add(line);

}

File.WriteAllLines(contactsFilePath, lines);

}

}