**NAME YABSIRA SOLOMON ID ATE/0705/11**

1. **How to reverse a string?**

public static stringReverseString3b(string str) {

    char[] chars = new char[str.Length];

    for (int i = 0, j = str.Length - 1; i <= j; i++, j--) {

        chars[i] = str[j];

        chars[j] = str[i];

    }

    return new string(chars);

}

1. **How to find if the given string is a palindrome or not?**

using System;

  public class PalindromeExample

   {

     public static void Main(string[] args)

      {

          int n,r,sum=0,temp;

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

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

          temp=n;

          while(n>0)

          {

           r=n%10;

           sum=(sum\*10)+r;

           n=n/10;

          }

          if(temp==sum)

           Console.Write("Number is Palindrome.");

          else

           Console.Write("Number is not Palindrome");

    }

  }

1. **How to reverse the order of words in a given string?**

{

    class Program

    {

        static void Main(string[] args)

        {

            Console.ForegroundColor = ConsoleColor.White;

            Console.WriteLine("Eneter the String:");

            Console.ForegroundColor = ConsoleColor.Yellow;

            string s=Console.ReadLine();

            string [] a = s.Split(' ');

             Array.Reverse(a);

             Console.ForegroundColor = ConsoleColor.Red;

             Console.WriteLine("Reverse String is:");

             for(int i=0;i<=a.Length-1;i++)

                {

                Console.ForegroundColor = ConsoleColor.White;

                Console.Write(a[i]+""+' ');

                }

            Console.ReadKey();

          }

        }

}

1. **How to count the occurrence of each character in a string?**

{

class Program

{

static void Main(string[] args)

{

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

string message = Console.ReadLine();

//Remove the empty spaces from the message

message = message.Replace(" ", string.Empty);

while (message.Length > 0)

{

Console.Write(message[0] + " : ");

int count = 0;

for (int j = 0; j < message.Length; j++)

{

if (message[0] == message[j])

{

count++;

}

}

Console.WriteLine(count);

message = message.Replace(message[0].ToString(), string.Empty);

}

Console.ReadKey();

}

}

}

1. **How to find all possible substrings of a given string?**

public static void PossibleUniqueSubString(string input)

{

    //declare the string variable

    string final = "";

    //Get a string of possible substrings with out uniqueness

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

    {

        string str = "";

        for (int j = i; j < input.Length; j++)

        {

            str += input[j];

            final += str + ",";

        }

    }

    //Remove the ending comma from the string

    final = final.Remove(final.Length - 1, 1);

    //Get an array after Spilt the string on basis of comma

    string[] arr = final.Split(',');

    //Get the distinct array

    arr = arr.Distinct().ToArray();

    //Print the array

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

        Console.Write(arr[i]+" ");

 }

### Find the second largest integer in an array using only one loop?

static void Main(string[] args)

        {

            int n, i, j = 0, largest, secondLargest;

            int[] arr1 = new int[50];

            Console.Write("\n\nFind the second largest element in an array :\n");

            Console.Write("-----------------------------------------\n");

            Console.Write("Input the size of array : ");

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

            /\* Stored values into the array\*/

            Console.Write("Input {0} elements in the array :\n", n);

            for (i = 0; i < n; i++)

            {

                Console.Write("element - {0} : ", i);

                arr1[i] = Convert.ToInt32(Console.ReadLine());

            }

            /\* find location of the largest element in the array \*/

            largest = 0;

            for (i = 0; i < n; i++)

            {

                if (largest < arr1[i])

                {

                    largest = arr1[i];

                    j = i;

                }

            }

            /\* ignore the largest element and find the 2nd largest element in the array \*/

            secondLargest = 0;

            for (i = 0; i < n; i++)

            {

                if (i == j)

                {

                    i++;  /\* ignoring the largest element \*/

                    i--;

                }

                else

                {

                    if (secondLargest < arr1[i])

                    {

                        secondLargest = arr1[i];

                    }

                }

            }

            Console.Write("The Second largest element in the array is :  {0} \n\n", secondLargest);

        Console.ReadKey();

        }

### Convert a two-dimensional array to a one-dimensional array?

using System;

namespace LogicalPrograms

{

class Program

{

static void Main(string[] args)

{

//Creating a 2d Array with 2 rows and three columns

int[,] int2DArray = new int[2, 3];

Console.Write("Enter 2D Array Elements : ");

for (int i = 0; i < 2; i++)

{

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

{

int2DArray[i, j] = Convert.ToInt32(Console.ReadLine());

}

}

int index = 0;

//Getting the no of rows of 2d array

int NoOfRows = int2DArray.GetLength(0);

//Getting the no of columns of the 2d array

int NoOfColumns = int2DArray.GetLength(1);

//Creating 1d Array by multiplying NoOfRows and NoOfColumns

int[] OneDimensionalArray = new int[NoOfRows \* NoOfColumns];

//Assigning the elements to 1d Array from 2d array

for (int y = 0; y < NoOfColumns; y++)

{

for (int x = 0; x < NoOfRows ; x++)

{

OneDimensionalArray[index] = int2DArray[x, y];

index++;

}

}

//Printing the 1d array elements

Console.WriteLine("1D Array Elements : ");

foreach (int item in OneDimensionalArray)

{

Console.Write(item + " ");

}

Console.ReadKey();

}

}

}

### How to find the angle between hour and minute hands of a clock at any given time?

using System;

namespace LogicalPrograms

{

class Program

{

static void Main(string[] args)

{

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

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

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

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

double hourInDegrees = (hours \* 30) + (minutes \* 30.0 / 60);

double minuteInDegrees = minutes \* 6;

double diff = Math.Abs(hourInDegrees - minuteInDegrees);

if (diff > 180)

{

diff = 360 - diff;

}

Console.WriteLine($"Angle between {hours} hour and {minutes} minute is {diff} degrees");

Console.ReadKey();

}

}

}

**Conceptual Questions**

1. **What is an Object?**

Is an instance of a class that is created dynamically. Object is also a keyword that is an alias for the predefined type System and object is created from a class.

1. **What is Encapsulation?**

The wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In a different way, encapsulation is a protective shield that prevents the data from being accessed by the code outside this shield. Is a process of binding the [data members](https://www.tutlane.com/tutorial/csharp/csharp-variables-with-examples) and [member functions](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) into a single unit.

1. **What is Abstraction**

Is the process of hiding certain details and showing only essential information to the user. Is an important part of object oriented programming. It means that only the required information is visible to the user and the rest of the information is hidden.

1. **Which are Access Specifiers?**

Access modifiers and specifiers are keywords (private, public, internal, protected and protected internal) to specify the accessibility of a type and its members.

1. **What is Inheritance?**

Is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which is defined in other class.

Is a feature of object-oriented programming languages that allows you to define a base class that provides specific functionality (data and behavior) and to define derived classes that either inherit or override that functionality.

1. **How can you implement multiple inheritance in C#?**
2. **Are private class members inherited to the derived class?**

No private members of the base-class are accessible within the derived-class and to the instances of derived-class.

1. **What is Polymorphism?**

Provides the ability to a class to have multiple implementations with the same name. It is one of the core principles of Object Oriented Programming after encapsulation and inheritance. The ability of objects of different types to provide a unique interface for different implementations of methods. It is usually used in the context of late binding, where the behavior of an object to respond to a call to its method members is determined based on object type at run time.

1. **What is method overloading?**

The ability to redefine a function in more than one form

1. **When and why to use method Overloading?**

When we actually do need multiple methods with different parameters, but the methods do the same thing. That is, don't use overloading if the multiple methods perform different tasks.

Why we use it is that it increases the readability of the program because you don't need to use different names for same action.

1. **What is method overriding?**

 Is a technique that allows the invoking of functions from another class (base class) in the derived class. Used to achieve runtime polymorphism. It enables you to provide specific implementation of the method which is already provided by its base class.

1. **What is Constructor?**

A special method of the class which gets automatically invoked whenever an instance of the class is created.

1. **Describe some of the key points regarding the Constructor.**

Is a member of a class. It is a method in the class which gets executed when a class object is created. Usually we put the initialization code in the constructor. The name of the constructor is always is the same name as the class.

1. **What is Private Constructor?**

Private constructor is a special instance constructor which is used in a class that contains static member only. It is used to stop object creation of a class.

1. **Can you create object of class with private constructor in C#?**

No, object of a class having private constructor cannot be instantiated from outside of the class.

1. **What is the use of private constructor in C#?**

* It’s used to stop object creation of a class.
* It’s used to stop a class to be inherited.
* It’s used in singleton design patterns, to make sure that the only one instance of a class can ever be created.

1. **What is the use of static constructor in C#?**

Used to initialize any static data, or to perform a particular action that needs to be performed only once.

Used to initialize static data members as soon as the class is referenced the first time

1. **What is Destructor?**

Methods inside the class used to destroy instances of that class when they are no longer needed. Used to destruct instances of classes.

1. **What is Namespaces?**

Is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

Used in C# to organize and provide a level of separation of codes.

1. **What are Virtual, Override, and New keywords in C#?**

Virtual keyword is used for generating a virtual path for its derived classes on implementing method overriding. The Virtual keyword is used within a set with an override keyword.

Override keyword is used in the derived class of the base class in order to override the base class method. The Override keyword is used with the virtual keyword

New keyword is also used for polymorphism but in the case of method overriding. We can say that we are changing what the base class does for the derived class..

1. **What is the difference between Struct and Class in C#?**

A [**class**](https://www.geeksforgeeks.org/c-class-and-object/)is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods(member function which defines actions) into a single unit.

**A**[**structure**](https://www.geeksforgeeks.org/c-structures-set-1/)is a collection of variables of different data types under a single unit. It is almost similar to a class because both are user-defined data types and both hold a bunch of different data types.

* Structs are value types while classes are reference types.
* Structs can be instantiated without using a new operator.

1. **What is Interface?**

is a completely "**abstract class**", which can only contain abstract methods and properties (with empty bodies):

1. **Why to use Interfaces in C#?**

include behavior from multiple sources in a class. That capability is important in C# because the language doesn't support multiple inheritance of classes.

1. **What is implicit interface implementation?**

Most regular or obvious way to implement members of an interface. Here we don't specify the interface name of the members and implement implicitly. The method can be declared at any interface (s) the class implements.

1. **What is explicit interface implementation?**

Another way to implement members of an interface. Here we need to specify the interface name of the members. The following example explains that.

1. **What is Abstract class?**

Is a special type of class that cannot be instantiated. An abstract class is designed to be inherited by subclasses that either implement or override its methods. In other words, abstract classes are either partially implemented or not implemented at all.

1. **Describe Abstract class in detail.**

An abstract class is an incomplete class or special class we can't be instantiated. The purpose of an abstract class is to provide a blueprint for derived classes and set some rules what the derived classes must implement when they inherit an abstract class. We can use an abstract class as a base class and all derived classes must implement abstract definitions. An abstract method must be implemented in all non-abstract classes using the override keyword. After overriding the abstract method is in the non-Abstract class. We can derive this class in another class and again we can override the same abstract method with it.

1. **What is the difference between Abstraction and Encapsulation?**

* Abstraction is the process or method of gaining the information. While encapsulation is the process or method to contain the information.
* We can implement abstraction using abstract class and interfaces. Whereas encapsulation can be implemented using by access modifier i.e. private, protected and public
* The objects that help to perform abstraction are encapsulated. Whereas the objects that result in encapsulation need not be abstracted

1. **Can Abstract class be sealed in C#?**

No, subclass of an abstract class can only be instantiated if it implements all of the abstract methods of its superclass. Such classes are called concrete classes to differentiate them from abstract classes.

1. **Can abstract class have Constructors in C#?**

Yes, an abstract class can have a constructor. In general, a class constructor is used to initialize fields. Along the same lines, an abstract class constructor is used to initialize fields of the abstract class.

1. **Can you declare abstract methods as private in C#?**

Private methods are not polymorphic (you cannot inherit them), so it makes no sense to make a private method abstract. Making a method abstract means you'd have to override and implement it in a subclass

1. **Can abstract class have static methods in C#?**

Yes, abstract class can have Static Methods. The reason for this is Static methods do not work on the instance of the class, they are directly associated with the class itself.

1. **Does Abstract class support multiple Inheritance?**

No, cannot be inherited by structures. It can contain constructors or destructors. It can implement functions with non-Abstract methods. It cannot support multiple inheritance

1. **Abstract class must have only abstract methods. Is it true or false?**
2. **When do you use Abstract Class?**

We use abstract class at the time of inheritance. A user must use the override keyword before the method which is declared as abstract in child class, the abstract class is used to inherit in the child class. An abstract class cannot be inherited by structures. It can contain constructors or destructors.

1. **Why can Abstract class not be Instantiated?**

Because the sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited.

1. **Which type of members can you define in an Abstract class?**

 To create classes and class members that are incomplete and must be implemented in a derived class.

1. **What is Operator Overloading?**

Operator overloading gives the ability to use the same operator to do various operations. It provides additional capabilities to C# operators when they are applied to user-defined data types

1. **Is it possible to restrict object creation in C#?**

Yes, we can limit the number of object creation of class in C# using the static variable.

1. **Can you inherit Enum in C#?**

No. it is not possible. Enum can not inherit in derived class because by default Enum is sealed.

1. **Is it possible to achieve Method extension using Interface?**

Yes, most of Linq is built around interface extension methods.

1. **Is it possible that a Method can return multiple values at a time?**

No, we can't return multiple values from a function in C# However, there are a couple alternatives: we can return an array of type object with the multiple values we want in it.

1. **What is Constant?**

Constants are fields whose values are set at compile time and can never be changed. Use constants to provide meaningful names instead of numeric literals ("magic numbers") for special values.

1. **What is Read only?**

Read only keyword is a modifier that can be used in four contexts: In a field declaration, read only indicates that assignment to the field can only occur as part of the declaration or in a constructor in the same class.

A read only field can't be assigned after the constructor exits.

1. **What is Static?**

 Static means something which cannot be instantiated. You cannot create an object of a static class and cannot access static members using an object. C# classes, variables, methods, properties, operators, events, and constructors can be defined as static using the static modifier keyword.

1. **What is Static Read Only?**

Static Read only type variable's value can be assigned at runtime or assigned at compile time and changed at runtime. But this variable's value can only be changed in the static constructor. And cannot be changed further.