**Lab Guide**

**Create two tables Department and Employee with relationship and implement all the joins**

**Solution**

Use Master

Create Database OrganizationDB

use OrganizationDB

Create table Department

(

Deptid int Primary Key identity(1000,100) ,

DeptName varchar(50) not null,

Hod varchar(50) not null

)

CREATE TABLE Employee(

EmployeeID int IDENTITY (1,1) NOT NULL,

FirstName nvarchar(50) NOT NULL,

LastName nvarchar(50) NOT NULL,

DepartmentID int NULL,

CONSTRAINT PK\_EmployeeID PRIMARY KEY(EmployeeID),

CONSTRAINT FK\_Employee\_Department FOREIGN KEY(DepartmentID)

REFERENCES Department(Deptid)

)

//when we enter only partial data, colnames in mandatory

// do not enter data for identity column

insert into Department(DeptName,Hod) Values('Sales','Samatha')

insert into Department(DeptName,Hod) Values('Accounts','Priyanka')

insert into Department(DeptName,Hod) Values('Marketing','Smruthi')

insert into Department(DeptName,Hod) Values('IT','Rekha')

insert into Department(DeptName,Hod) Values('Testing','Raghu')

insert into Employee(FirstName,LastName,DepartmentID) values('samatha','Ramakrishna',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Samadrita','Chaterjee',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Supriya','Karn',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Margana','Neelima',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Rimpa','Satpathi',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Krishita','Viroja',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Priyanka','Kanubai Sagar',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Shruti','Kumari',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Smruthi','KalpanaDutta',Null)

insert into Employee(FirstName,LastName,DepartmentID) values('Gadde','Apoorva',Null)

Select \* Department

Select \* Employee

//retireves only common data

select e1.EmployeeID,e1.FirstName,d1.Depid,d1.DeptName

from Employee e1 Join Department d1

on d1.Depid = e1.DepartmentID

select e1.EmployeeID,e1.FirstName,d1.Depid,d1.DeptName

from Employee e1 left outer Join Department d1

on d1.Depid = e1.DepartmentID

select e1.EmployeeID,e1.FirstName,d1.Depid,d1.DeptName

from Employee e1 right outer Join Department d1

on d1.Depid = e1.DepartmentID

select e1.EmployeeID,e1.FirstName,d1.Depid,d1.DeptName

from Employee e1 Full outer Join Department d1

on d1.Depid = e1.DepartmentID

**Complete the class Account and AccountDetails as per the below requirement**

**Problem Statement - Account Details**

**class Account :**

Create the following instance/static members:

accountNo : int

balance : double

accountType : string

counter :int static

Define parameterized constructor with two parameters to initialize balance and accountType. accountNo should be initialized by incrementing counter.

* Implement the below operations:
* void depositAmount(double amount)
* To add amount to account balance
* void printAccountDetails()
* To display account details as per format given in Example Section

**class AccountDetails :**

* Create GetData() method and follow the below instructions.
* Accept balance, account type and amount as input for two account objects from Console(Refer Example section for input format)
* create first object using the input data and display account details
* Deposit amount using the input data and display the new account balance
* create second account object using the input data and display account details.
* Set account balance to new balance using input data and display the new account balance

Note : Don't Implement the Main method

**Example**

Sample Input:

100.5

Savings

25.5 // balance type amount for first account

200

Current

50.5 // balance type amount for second account

Expected Output:

[Acct No : 1, Type : Savings, Balance : 100.5]

New Balance : 126.0

[Acct No : 2, Type : Current, Balance : 200.0]

New Balance : 250.5

Sample Input:

0

Current

100

0

Current

50

Expected Output:

[Acct No : 1, Type : Current, Balance : 0.0]

New Balance : 100.0

[Acct No : 2, Type : Current, Balance : 0.0]

New Balance : 50.0

**Instructions**

* Do not change the provided class/method names unless instructed
* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

**Write a Program in C# which will convert Centigrade to Fahrenheit**

**Solution:**

using System;

class TemperatureConverter

{

    static void Main()

    {

        Console.WriteLine("Temperature Converter: Celsius to Fahrenheit");

        // Input temperature in Celsius

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

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

        // Convert Celsius to Fahrenheit

        double fahrenheit = CelsiusToFahrenheit(celsius);

        // Display the result

        Console.WriteLine($"Temperature in Fahrenheit: {fahrenheit} °F");

    }

    // Function to convert Celsius to Fahrenheit

    static double CelsiusToFahrenheit(double celsius)

    {

        // Formula: (°C × 9/5) + 32

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

    }

}

**Write a Program in C# which will convert Centigrade to Fahrenheit**

Create a function PayRate which takes rate as input parameter and returns a table with EmployeeID, RateChangeDate, Rate. PayFrequency, ModifiedDate as columsn. All the employees whose rate is greater the rate parameter. Use Adventureworks database and use Employee Payment History . Use Sql functions with Multivalued Table function

Solution

CREATE FUNCTION PayRate (@rate money)

                                RETURNS @table TABLE

                                (EmployeeID int NOT NULL,

                                RateChangeDate datetime NOT NULL,

                                Rate money NOT NULL,

                                PayFrequency tinyint NOT NULL,

                                ModifiedDate datetime NOT NULL)

                                AS

                                BEGIN

                                           INSERT @table

                                           SELECT \* FROM

                             HumanResources.EmployeePayHistory

                                           WHERE Rate > @rate

                                RETURN

                                END

SELECT \* FROM PayRate(45)

**Write a class called Grade with m1,m2,m3,m4,m5,average as integers. GetDetails which accepts 5 subject marks private string CalculateGrade(average) check the average and return grade if average >=80 and <=100 return first if average >=50 and <80 return second if average >=30 and <50 return third if average >=0 and <30 return fail else return invalid marks write DisplayDetails() and call CalculateGrade in DisplayDetails()**

Solution:

using System;

class Grade

{

    private int m1, m2, m3, m4, m5;

    private int average;

    // Method to get subject marks

    public void GetDetails()

    {

        Console.Write("Enter marks for subject 1: ");

        m1 = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter marks for subject 2: ");

        m2 = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter marks for subject 3: ");

        m3 = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter marks for subject 4: ");

        m4 = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter marks for subject 5: ");

        m5 = Convert.ToInt32(Console.ReadLine());

    }

    // Method to calculate grade based on average

    private string CalculateGrade(int average)

    {

        if (average >= 80 && average <= 100)

        {

            return "First";

        }

        else if (average >= 50 && average < 80)

        {

            return "Second";

        }

        else if (average >= 30 && average < 50)

        {

            return "Third";

        }

        else if (average >= 0 && average < 30)

        {

            return "Fail";

        }

        else

        {

            return "Invalid marks";

        }

    }

    // Method to display details and call CalculateGrade

    public void DisplayDetails()

    {

        average =

Write a Program Called Login with username as string and password as string . Accept username and password in GetDetails() .   ValidateUser(username,password) return true if valid user return false if invalid user. Call ValidateUser in DisplayResult . if validuser display login successful else display login failed

**Solution**

using System;

class LoginProgram

{

    private string username;

    private string password;

    // Method to accept username and password

    public void GetDetails()

    {

        Console.Write("Enter username: ");

        username = Console.ReadLine();

        Console.Write("Enter password: ");

        password = Console.ReadLine();

    }

    // Method to validate user

    private bool ValidateUser(string username, string password)

    {

        // For demonstration purposes, let's consider a simple validation

        // You can replace this with your own logic, such as checking against a database

        return username == "admin" && password == "admin123";

    }

    // Method to display result based on validation

    public void DisplayResult()

    {

        bool isValidUser = ValidateUser(username, password);

        if (isValidUser)

        {

            Console.WriteLine("Login successful!");

        }

        else

        {

            Console.WriteLine("Login failed. Invalid username or password.");

        }

    }

    static void Main()

    {

        LoginProgram loginProgram = new LoginProgram();

        // Get user details

        loginProgram.GetDetails();

        // Validate user and display result

        loginProgram.DisplayResult();

    }

}

John asked for a puzzle from one of his friends. He has given a string and he has to decode the given string according to the set of rules

1. Reverse each word of the space-separated string.
2. Eliminate palindrome word.

Palindrome words are those words that can be read the same from either side. For example – “aba” is the same as the reverse of “aba”; Therefore, it is a palindrome.

**Input Format**

* First line contains the value of **N**, no. of queries(String).
* Next **N** lines contains string.

**Output Format**

* For each query print the decoded string in a newline.

**Constraints**

* 1 <= **N** <= 100
* 1 <=**length of s** <= 1000

**Sample Input**

2

i love my country

she is madam

**Sample Output**

evol ym yrtnuoc

ehs si

**Explanation**

Query 1 -

* After reversing each word -**I evol ym yrtnuoc**.
* After eliminating palindrome words - **evol ym yrtnuoc**

Query 2 -

* After reversing each word - **ehs is madam**
* After eliminating palindrome words - **ehs is**

Solution :

using System;

using System;

using System.Linq;

class Program

{

    static void Main(string[] args)

    {

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

        for (int i = 0; i < N; i++)

        {

            string input = Console.ReadLine();

            string[] words = input.Split(' ');

            string decodedString = DecodeString(words);

            Console.WriteLine(decodedString);

        }

    }

    static string DecodeString(string[] words)

    {

        string decoded = "";

        foreach (string word in words)

        {

            string reversed = ReverseWord(word);

            if (!IsPalindrome(reversed))

            {

                decoded += reversed + " ";

            }

        }

        return decoded.Trim();

    }

    static string ReverseWord(string word)

    {

        char[] charArray = word.ToCharArray();

        Array.Reverse(charArray);

        return new string(charArray);

    }

    static bool IsPalindrome(string word)

    {

        int left = 0;

        int right = word.Length - 1;

        while (left < right)

        {

            if (word[left] != word[right])

            {

                return false;

            }

            left++;

            right--;

        }

        return true;

    }

}

**Write a C# Program Which accepts the name and age and display whether the person is eligible to vote**

Solution

using System;

class VotingEligibilityChecker

{

static void Main()

{

Console.WriteLine("Voting Eligibility Checker");

// Input name

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

string name = Console.ReadLine();

// Input age

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

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

// Check eligibility

bool isEligible = CheckVotingEligibility(age);

// Display the result

Console.WriteLine($"{name}, you are {(isEligible ? "eligible" : "not eligible")} to vote.");

}

// Function to check voting eligibility

static bool CheckVotingEligibility(int age)

{

// Voting age in most countries is 18

const int votingAge = 18;

return age >= votingAge;

}

}

**Find Maximum and Minimum Age**

Complete the main method to accept the age of n students and find the maximum and minimum age.

The first input is the number n representing the number of age values you need to enter as integers

Followed by the age values in a separate line.

The output should display as shown below in sample input /output.

**Following requirements should be taken care in the program.**

1. Input should be taken through Console
2. Program should print the output as described in the Example Section below
3. The number n representing the number of students should be allowed in the range of 1 to 20
4. If n is entered less than 1 or more than 20 , it should print message as INVALID\_INPUT.

**Example**

Sample Input 1:

5

34

56

12

﻿89

43

Sample Ouptut 1:

MIN=12

MAX=89

Sample Input 2:

25

Expected Output:

INVALID\_INPUT

Sample Input 3:

8

78

44

23

65

45

9

23

39

Expected Output:

MIN=9

MAX=78

**Instructions**

* Do not change the provided class/method names unless instructed
* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

using System;

namespace LearnCsharp

{

class FindMaxMinAge{

    public static void Main(string[] args) {

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

        if (n < 1 || n > 20)

        {

            Console.WriteLine("INVALID\_INPUT");

        }

        else

        {

            int[] ages = new int[n];

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

            {

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

            }

            int min = ages[0];

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

            {

                if (ages[i] < min)

                    min = ages[i];

            }

            Console.WriteLine("MIN=" + min);

            int max = ages[0];

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

            {

                if (ages[i] > max)

                    max = ages[i];

            }

            Console.WriteLine("MAX=" + max);

        }

    }

}

}

**Instructions**

* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Example**

Sample Input:

Ravi

Somebody

Tanvir

Ramesh

Nobody

Ani

Nobody

Vishwanath

Somebody

Nitin

Expected output :

Ani

Nitin

Ramesh

Ravi

Tanvir

Vishwanath﻿

﻿

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

using System;

using System.Collections;

namespace LearnCsharp

{

class NamesWithArrayList

{

    public static void Main(string[] args)

    {

//Update the code below

        ArrayList alObj;

            alObj = new ArrayList();

int max=10;

string item="";

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

{

    item=Console.ReadLine();

    if(item != null)

    {

        alObj.Add(item);

    }

}

          for(int j=0;j<alObj.Count;j++)

           {

               if(alObj[j].ToString()=="Somebody".Trim() || alObj[j].ToString()=="Nobody".Trim())

               alObj.Remove(alObj[j]);

           }

            alObj.Sort();

            foreach (var item1 in alObj)

            {

                Console.WriteLine(item1);

            }

        }

}

**Your task here is to implement a C# code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields, and methods unless**

**Specifications:**

class definitions:

  class Book:

      method definitons:

        Name: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Price: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Author: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

**Year**: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Book(**string** **name**, **string** price, **string** author, **string** **year**) : **constructor**

            visibility: **public**

        Ser(**List**<Book> books) : method **to** implement serialization

**return** **type**: stream

            visibility: **public**

**return**: stream(serialized **list** **in** binary **format**)

        Deser(FileStream s): method **to** implement deserialization

**return** **type**: **List**

            visibility: **public**

**return**: deserialized **list**

**main**(**String** args[]): method **of** **type** **static** void

**List**<Book> **list**: **List**

                s: FileStream

                method calls:

                    Ser(**list**)

                    Deser(s)

**Task:**

Create a Book class with**string Name, string Price, string Author, string Year attributes,** your task is to implement the below given methods in order to perform serialization and deserialization.

* Define getter setter method using **Auto Implementation Property**
* Define parameterized constructor.
* Implement **Ser(List<Book> books)** method to serialize List<Book>. The serialization, which takes place should be done by sending the Serialize message to the BinaryFormatter object and serialize it to the file called bks.txt.(The serialization relies on a binary stream, represented by an instance of class Filestream)
* Implement **Deser(FileStream s)** method to deserialize the list from the file .

**Note:**

The class which needs to be serialized needs to have the [Serializable] attribute.

**IMPORTANT:**

* If you want to test your program you can implement a Main() function given in the stub and you can use RUN CODE to test your Main() provided you have made valid function calls with valid data required.

**Sample Input**

Book first = **new** Book( "Alchemist", 175, "Paulo Coelho", 1988 );

**mentioned otherwise.**

**Specifications:**

class definitions:

  class Book:

      method definitons:

        Name: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Price: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Author: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

**Year**: Implement getter setter method (**use** **Auto** Implementation Property)

**return** **type**: **string**

            visibility: **public**

        Book(**string** **name**, **string** price, **string** author, **string** **year**) : **constructor**

            visibility: **public**

        Ser(**List**<Book> books) : method **to** implement serialization

**return** **type**: stream

            visibility: **public**

**return**: stream(serialized **list** **in** binary **format**)

        Deser(FileStream s): method **to** implement deserialization

**return** **type**: **List**

            visibility: **public**

**return**: deserialized **list**

**main**(**String** args[]): method **of** **type** **static** void

**List**<Book> **list**: **List**

                s: FileStream

                method calls:

                    Ser(**list**)

                    Deser(s)

**Task:**

Create a Book class with**string Name, string Price, string Author, string Year attributes,** your task is to implement the below given methods in order to perform serialization and deserialization.

* Define getter setter method using **Auto Implementation Property**
* Define parameterized constructor.
* Implement **Ser(List<Book> books)** method to serialize List<Book>. The serialization, which takes place should be done by sending the Serialize message to the BinaryFormatter object and serialize it to the file called bks.txt.(The serialization relies on a binary stream, represented by an instance of class Filestream)
* Implement **Deser(FileStream s)** method to deserialize the list from the file .

**Note:**

The class which needs to be serialized needs to have the [Serializable] attribute.

**IMPORTANT:**

* If you want to test your program you can implement a Main() function given in the stub and you can use RUN CODE to test your Main() provided you have made valid function calls with valid data required.

**Sample Input**

Book first = **new** Book( "Alchemist", 175, "Paulo Coelho", 1988 );

**Write a Program that behaves as prescribed in the below problem statement.**

Write a program to initialize a single dimensional array of any size **with** integer **values** . Display the **complete** array **content** **with** **count** **of** the numbers which **are** divisible **by** 5 **to** the end.

**Size** **of** the array **is** **first** **input** **to** the program followed **by** the elements **of** the array.

* Assume all input values are >= 5
* Take input/output as specified
* Print the expected output using the expected logic/algorithm/data
* Code is structured correctly and according to the problem statement

**Instructions**

* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Example**

Sample Input :

7       //Size of the array

5

﻿10

15

16

﻿1

10

﻿21

Expected Output :

Count of elements divide by 5: 4

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

Solution

using System;

namespace LearnCsharp

{

 class CountOfDivideByFive

 {

     public static void Main(string[] args)

     {

   //Write Your Code Here

     int[] iArray;

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

            iArray = new int[Count];

            int iCount = 0;

            for (int iVal1 = 0; iVal1 < Count; iVal1++)

            {

                iArray[iVal1] = int.Parse(Console.ReadLine());

            }

            foreach (int item in iArray)

            {

                if ((item % 5) == 0)

                    iCount++;

                Console.WriteLine(item);

            }

            Console.WriteLine("Count of elements divide by 5: " + iCount);

     }

 }

}

**Write a function sumprimes(ls) that takes as input a list of integers ls and returns the sum of all the prime numbers in ls.**

**nput:**

The input has a list of values separated by space.

**Output:**

A single number representing sum of all prime numbers in the given list.

**Sample Input 1:**

3 3 1 13

**Sample Output 1:**

19

Solution:

using System;

using System.Linq;

class Program

{

    static bool IsPrime(int num)

    {

        if (num < 2)

            return false;

        for (int i = 2; i <= Math.Sqrt(num); i++)

        {

            if (num % i == 0)

                return false;

        }

        return true;

    }

    static int SumPrimes(int[] numbers)

    {

        return numbers.Where(IsPrime).Sum();

    }

    static void Main()

    {

        Console.WriteLine("Enter a list of integers separated by space:");

        string input = Console.ReadLine();

        // Split the input string into an array of integers

        int[] numbers = input.Split(' ').Select(int.Parse).ToArray();

        // Calculate and display the sum of prime numbers

        int result = SumPrimes(numbers);

        Console.WriteLine($"Sum of prime numbers: {result}");

    }

}

**Write a Program which behaves as prescribed in the below problem statement**

**Create** a program **to** **store** **list** **of** marks **using** generic collection class.

The program should **allow** **storage** **of** int **values** **only** **and** it

should **sort** the **list** **of** marks **before** they **are** displayed **using**

DisplayGlist function.

The program should **ignore** non integer values.

* Take input/output as specified
* Print the expected output using the expected logic/algorithm/data
* Code is structured correctly and according to the problem statement

**Instructions**

* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Example**

sample Input:

51

88

92

56

xyz﻿

Expected Output :

﻿51

56

88

92

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

using System;

using System.Collections.Generic;

/\*

Question:

41. Create a program to store list of marks using generic collection class.   The program should allow storage of int values only and it should

sort the list of marks before they are displayed using DisplayGlist function.   The program should ignore non integer values.   Values are

passed to the program using command line arguments.

\*/

namespace LearnCsharp

{

class M7StudentMarks12

{

    public static void Main(string[] args)

    {

    //Complete/Update the code below

List<int> lMarks;

            lMarks = new List<int>();

            int iVal,iVa1=0;

            string[] marks=new string[100];

            int i=0;

            do

            {

                marks[i]=Console.ReadLine();

                if(marks[i] == null)

                break;

                if(int.TryParse(marks[i],out iVal) )

                lMarks.Add(iVal);

                i++;

            }while(marks[i] == null);

           /\* foreach (string sItem in args)

            {

                if (int.TryParse(sItem, out iVal))

                {

                iVal--;

                    lMarks.Add(iVa1);

                }

            } \*/

            lMarks.Sort();

            DisplayGlist(lMarks);

        }

        private static void DisplayGlist(List<int> lMarks)

        {

            foreach (int lObj in lMarks)

            {

                Console.WriteLine(lObj);

            }

        }

    }

}

**Your task here is to implement a C# code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.**

**Specifications:**

**class** **Source**:

   method definitons:

        Count(Dictionary<string, string> dict): get count of key/value pairs in Dictionary

**return** type: **int**

            visibility: **public**

      CheckKey(Dictionary<**int**, string> dict): Method to check **if** key 3 is available in Dictionary.

**return** type: **int**

         visibility: **public**

     Values(Dictionary<string, string> dict): Method to get the values in Dictionary

**return** type: string

         visibility: **public**

**Task:**

Given a class **Source,** your task here is to implement the below given methods:

* **Count(Dictionary<string, string> dict):**Method to get the **count** of key/value pairs in Dictionary
* **CheckKey(Dictionary<int, string> dict):** Method to check if key **3** is available in Dictionary. If key is present in Dictionary return **value** paired with key **3** else return **"Could not find the specified key."**
* **Values(Dictionary<string, string> dict):** Method to get the **values** in Dictionary

**IMPORTANT:**

* If you want to test your program you can implement a **Main()**method given in the stub and you can use **RUN CODE** to test your Main(), provided you have made valid function calls with valid data required.

using System;

using System.Collections;

using System.Linq;

using System.Collections.Generic;

class Source {

    public string CheckKey(Dictionary<int, string> dict){

    string result;

    if(dict.TryGetValue(3, out result))

    {

        return (result);

    }

    else

    {

      return ("Could not find the specified key.");

    }

    }

    // public bool CheckForPair(Dictionary<int, string> dict){

    // return (dict.Contains(new KeyValuePair<int,string>(1,"One")));

    // }

    public int Count(Dictionary<string, string> dict){

        return (dict.Count);

    }

    public string Values(Dictionary<string, string> dict){

    Dictionary<string, string>.ValueCollection valueColl=dict.Values;

    var str = "";

    foreach(string s in valueColl) {

          str += s + " ";

    }

    return str;

    }

}

**Write a Program which behaves as prescribed in the below problem statement mentioned otherwise.**

**Create** a program which can **save** **list** **of** messages **in** the stack **object** **and** same should be processed (display) **using** a **function** ProcessStack.Prgram takes the **input** **as**  **list** **of** messages **in** a single line, **and** displays the **all** the  messages,  **each** message **in** a separate line.

**Example**

sample Input:

"email from Ram at 10:10 am" "email from Ramesh at 10:15 am" "email from Rajan at 10:20 am" "email from Rakesh at 10:25 am"

Expected Output:

email **from** Rakesh at 10:25 am

email **from** Rajan at 10:20 am

email **from** Ramesh at 10:15 am

email **from** Ram at 10:10 am

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

**Hints**

* Use Split() method from the String class to process the input

Solution

using System;

using System.Collections;

namespace LearnCsharp

{

 class M7WorkingWithStack8

 {

     public static void Main(string[] args)

     {

   //Write Your Code Here

   string str=Console.ReadLine();

   string[] msgs=str.Split('"' );

            Stack sObj;

            sObj = new Stack();

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

{

 if(msgs[i].Length > 1)

            sObj.Push(msgs[i]);

}

            ProcessStack(sObj);

        }

        private static void ProcessStack(Stack qObj)

        {

            while (qObj.Count > 0)

            {

                Console.WriteLine(qObj.Pop().ToString());

            }

        }

 }

}

**Complete the main method to Accept n numbers and display the numbers in ascending order as output ,if n is even. If n is odd, then display the numbers in descending order**

**Following requirements should be taken care in the program.**

1. Input should be taken through Console
2. Program should print the output as described in the Example Section below
3. The first input n should represent the total number of values entered followed by the actual values to be sorted.
4. n should be within the range of 1 to 20 . If n is entered as less than 1 or more than 20 , it should show message as INVALID\_INPUT.

**Example**

Sample Input 1:

7

23

45

﻿67

﻿﻿97

65

34

74

Expected Output:

97 74 67 65 45 34 23

Sample Input 2:

6

77

44

﻿22

65

﻿28

﻿43

Expected Output2:

22 28 43 44 65 77

﻿

Sample Input 3:

0

Expected Output 3:

INVALID\_INPUT

Sample Input 4:

30

Expected Output 4:

INVALID\_INPUT

**Instructions**

* Do not change the provided class/method names unless instructed
* Ensure your code compiles without any errors/warning/deprecations
* Follow best practices while coding
* Avoid too many & unnecessary usage of white spaces (newline, spaces, tabs, ...), except to make the code readable
* Use appropriate comments at appropriate places in your exercise, to explain the logic, rational, solutions, so that evaluator can know them
* Try to retain the original code given in the exercise, to avoid any issues in compiling & running your programs
* Always test the program thoroughly, before saving/submitting exercises/project
* For any issues with your exercise, contact your coach

**Warnings**

* Take care of whitespace/trailing whitespace
* Trim the output and avoid special characters
* Avoid printing unnecessary values other than expected/asked output

Solution

using System;

namespace LearnCsharp

{

class Sortnumbers{

    public static void Main(string[] args) {

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

        if (n < 1 || n > 20)

        {

            Console.WriteLine("INVALID\_INPUT");

        }

        else

        {

            int[] arr = new int[n];

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

            {

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

            }

            if (n % 2 == 0)

            {

                Array.Sort(arr);

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

                {

                    Console.Write(arr[i] + " ");

                }

            }

            else

            {

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

                {

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

                    {

                        if (arr[i] > arr[j])

                        {

                            int temp;

                            temp = arr[i];

                            arr[i] = arr[j];

                            arr[j] = temp;

                        }

                    }

                }

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

                {

                    Console.Write(arr[i] + " ");

                }

            }

          }

    }

}

}

**Create a stored procedure called prcGetEmployeeDetails with EmpId as input parameter and DepName and ShiftId as output parameter. If the empid exists, retrieve Department Name, ShiftID form Department and EmployeeDepartmentHistory and return 1 else return 0.**

Call prcGetEmployeeDetails stored procedure in another stored procedure prcDisplayEmployeeStatus

Step1:

Use AdventureWorks2019

go

Step2:

CREATE PROCEDURE prcGetEmployeeDetail @EmpId int, @DepName char(50) OUTPUT, @ShiftId int OUTPUT

              AS

              BEGIN

              IF EXISTS(SELECT \* FROM HumanResources.Employee WHERE EmployeeID = @EmpId)

              BEGIN

                             SELECT @DepName = d.Name, @ShiftId = h.ShiftID

    FROM HumanResources.Department d JOIN

    HumanResources.EmployeeDepartmentHistory h

                             ON d.DepartmentID = h.DepartmentID

                             WHERE EmployeeID = @EmpId AND h.Enddate IS NULL

                             RETURN 0

              END

              ELSE

              RETURN 1

END

Step3:

              CREATE PROCEDURE prcDisplayEmployeeStatus @EmpId int

              AS

              BEGIN

     DECLARE @DepName char(50)

     DECLARE @ShiftId int

     DECLARE @ReturnValue int

     EXEC @ReturnValue = prcGetEmployeeDetail @EmpId,

   @DepName OUTPUT,@ShiftId OUTPUT

     IF (@ReturnValue = 0)

     BEGIN

        PRINT 'The details of an employee with ID: ' +

      convert(char(10), @EmpId)

        PRINT 'Department Name: ' + @DepName

              PRINT 'Shift ID: ' + convert( char(1), @ShiftId)

  SELECT ManagerID, Title FROM

              HumanResources.Employee

              WHERE EmployeeID = @EmpID

END

ELSE

  PRINT 'No records found for the given employee'

END

Step4:

EXEC prcDisplayEmployeeStatus 2