1. Write the C# code to display “<n> minute(s) ago" if difference between both dates is less than an hour and “<n> hour(s) ago" if the difference is less than a day and “<n> day(s) ago" if the difference is less than a month and the date if the difference is more than a month.

namespace Assignments

{

class Program

{

public static string getTimeStamp(DateTime currentDate, DateTime updateDate)

{

int YearDiff = currentDate.Year - updateDate.Year;

int MonthDiff = currentDate.Month - updateDate.Month;

int DayDiff = currentDate.Day - updateDate.Day;

int HourDiff = currentDate.Hour - updateDate.Hour;

int MinuteDiff = currentDate.Minute - updateDate.Minute;

int SecondDiff = currentDate.Second - updateDate.Second;

if (YearDiff > 1)

return $"File was updated {YearDiff} years ago";

else if (MonthDiff > 1)

return $"File was updated {MonthDiff} Months ago";

else if (DayDiff > 1)

return $"File was updated {DayDiff} Days ago";

else if (HourDiff > 1)

return $"File was updated {HourDiff} Hours ago";

else if (MinuteDiff > 1)

return $"File was updated {MinuteDiff} Minutes ago";

else

return $"File was updated {SecondDiff} Seconds ago";

}

static void Main(string[] args)

{

DateTime currentDate = DateTime.Now;

Console.Write("Enter Date (dd-mm-yyyy hh:mm:ss) : ");

DateTime oldDate = DateTime.Parse(Console.ReadLine());

Console.WriteLine("\n\n" + getTimeStamp(currentDate, oldDate));

Console.ReadKey();

}

}

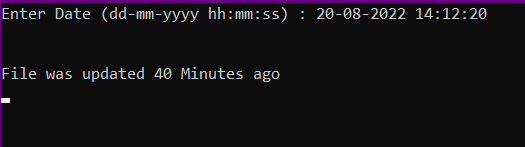
}

Text

Description automatically generated

Text

Description automatically generated



1. Write a program in C# Sharp to create and read last n number of lines of a file. Go to the editor.

namespace Assignments

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("How many lines you want to enter");

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

Console.WriteLine("Enter some text lines to enter in text file: \n");

string[] textToWrite = new string[LineNum];

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

{

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

textToWrite[i] = Console.ReadLine();

}

File.WriteAllLines(@"C:\Users\samsc\OneDrive\Desktop\Spektra C# Assigments\dummy.txt", textToWrite);

Console.WriteLine("Text written in the file successfully...\n\n");

Console.WriteLine("How many files you want to disply: ");

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

string[] GetText = File.ReadAllLines(@"C:\Users\samsc\OneDrive\Desktop\Spektra C# Assigments\dummy.txt");

Console.WriteLine("Here are the selected lines : \n");

foreach(string text in GetText.TakeLast(getLines))

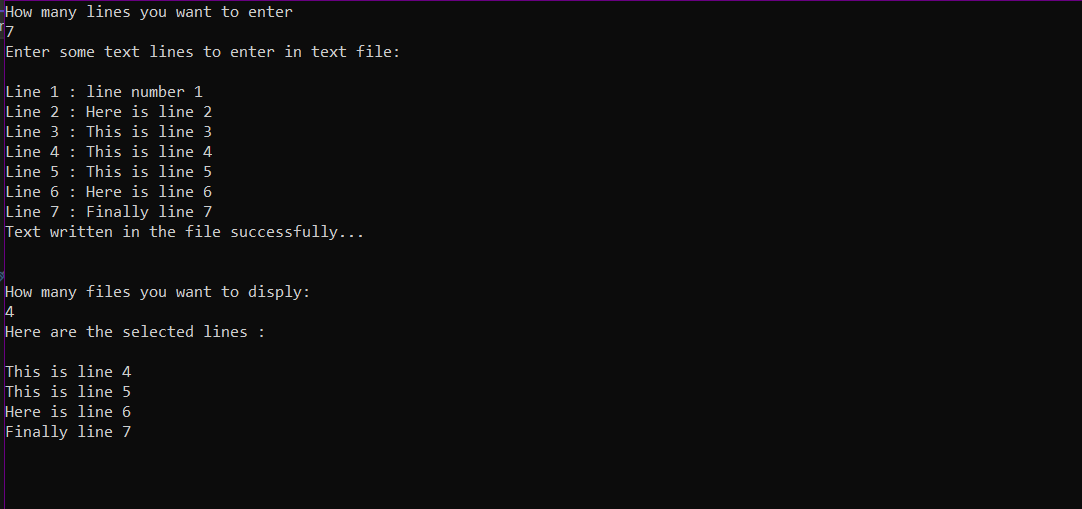
Console.WriteLine(text);

Console.ReadKey();

}

}

}



1. Write a function:

Class Solution {

public int solution(int[] A );

}

That, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

namespace Assignments

{

class Program

{

public static int Solution(int[] A, int len)

{

Array.Sort(A);

if (A[len] <= 0)

return 1;

else

{

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

{

if (A[i] > 0)

{

for (int j = 1; j < A[len]; j++)

{

if (!A.Contains(j))

{

return j;

}

}

}

}

}

return (A[len]+1);

}

static void Main(string[] args)

{

int[] IntArr = { -1, 2, -8, -3, -4, -5, 1, 3 };

foreach (int i in IntArr)

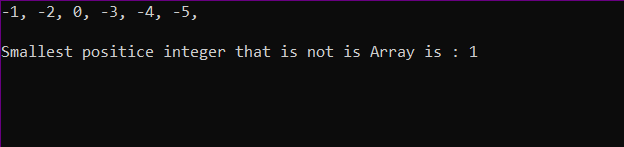
Console.Write(i + ", ");

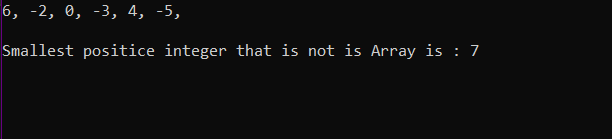
Console.WriteLine("\n\nSmallest positive integer that is not is Array is : " + Solution(IntArr, IntArr.Length-1));

Console.ReadKey();

}

}

}



1. Given six digits, find the earliest valid time that can be displayed on a digital clock (in 24-hour format) using those digits.

namespace Assignments

{

class Program

{

public static int[] Swap(int[] arr, int From, int To)

{

int temp;

for (int i = From; i > To; i--)

{

temp = arr[i];

arr[i] = arr[i-1];

arr[i-1] = temp;

}

return arr;

}

public static string TimeStamp(int[] arr)

{

string ClockTime = "";

for (int i = 0; i < 6; i++)

{

ClockTime += arr[i].ToString();

if (i == 1 || i == 3)

ClockTime += ":";

}

return ClockTime;

}

public static string Solution(int A, int B, int C, int D, int E, int F)

{

int[] ClockDigits = { A, B, C, D, E, F };

Array.Sort(ClockDigits);

if((ClockDigits[0] == 2 && ClockDigits[1] > 4) || ClockDigits[0] > 2)

{

return $"Not possible to convert {TimeStamp(ClockDigits)}, invalid time...";

}

if (ClockDigits[3] > 5 && ClockDigits[4] > 5 && ClockDigits[5] > 5)

{

if (ClockDigits[2] > 5 || ClockDigits[0] > 2)

{

return $"Not possible to convert {TimeStamp(ClockDigits)}, invalid time...";

}

if (ClockDigits[0] == 1)

{

ClockDigits = Swap(ClockDigits, 4, 1);

ClockDigits = Swap(ClockDigits, 4, 3);

return $"Minimum time is -> {TimeStamp(ClockDigits)}";

}

}

ClockDigits = Swap(ClockDigits, 4, 3);

return $"Minimum time is -> {TimeStamp(ClockDigits)}";

}

static void Main(string[] args)

{

Console.WriteLine("Enter 6 single digits for digital clock: ");

Console.Write("Digit 1 : ");

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

Console.Write("Digit 2 : ");

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

Console.Write("Digit 3 : ");

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

Console.Write("Digit 4 : ");

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

Console.Write("Digit 5 : ");

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

Console.Write("Digit 6 : ");

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

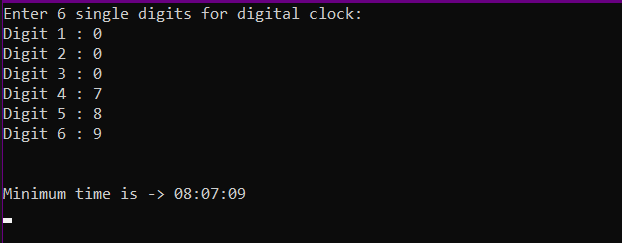
Console.WriteLine("\n\n" + Solution(a, b, c, d, e, f));

Console.ReadKey();

}

}

}



Text

Description automatically generated

Text

Description automatically generated

1. Reverse the ordering of words in a String

namespace Assignments

{

class Program

{

static void Main(string[] args)

{

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

string? Str = Console.ReadLine();

string[] Words = Str.Split(" ");

Console.Write("\n\nReversed String of words : ");

foreach(string word in Words.Reverse())

Console.Write(word + " ");

Console.ReadKey();

}

}

}

Text

Description automatically generated