Day 5 – Assignment Report

# Count Repeating Digits

## Code

namespace Day5Tasks

{

internal class Program

{

static int countNumbersWithRepeating(int[] array, bool singleDigit = true) {

int count = 0;

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

{

string num = array[i].ToString();

if (singleDigit && num.Length == 1) {

continue;

}

char prev = num[0];

bool flag = true;

for (int j = 1; j < num.Length; j++)

{

if (prev != num[j]) {

flag = false; break;

}

prev = num[j];

}

if (flag)

{

count++;

}

}

return count;

}

static void Main(string[] args)

{

int[] arr = { 111, 222, 333, 1, 19, 0, 12 };

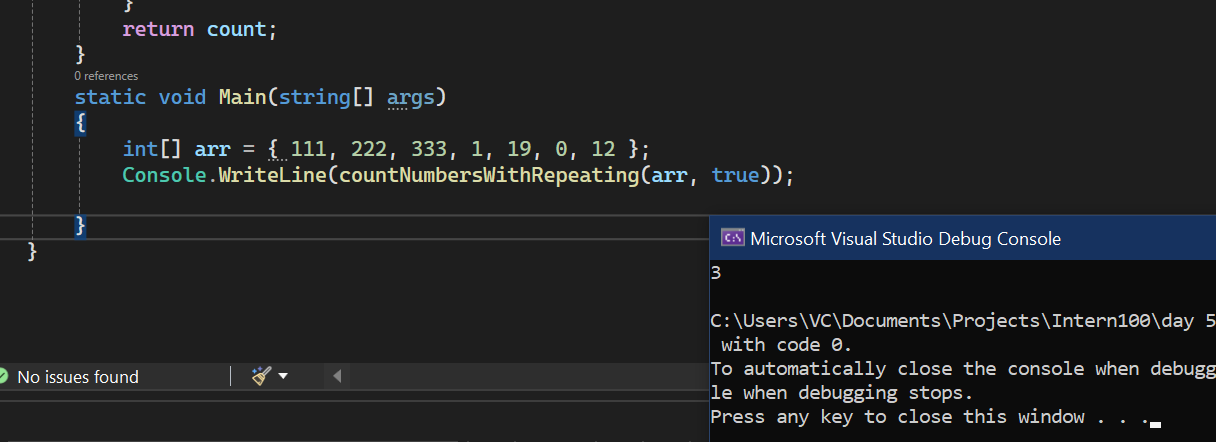
Console.WriteLine(countNumbersWithRepeating(arr, false));

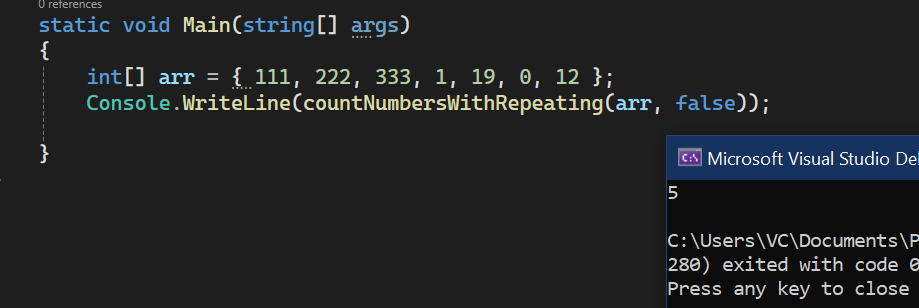
}

}

}

## Outputs





# Employee Tracker

## Code

### EmployeeTracker.program

using EmployeeTrackerModelLibrary;

using System.Runtime.ExceptionServices;

namespace EmployeeTracker

{

internal class Program

{

Employee[] employees;

public Program()

{

employees = new Employee[3];

}

void PrintMenu()

{

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

Console.WriteLine("2. Print Employees");

Console.WriteLine("3. Search Employee by ID");

Console.WriteLine("4. Update Employee by ID");

Console.WriteLine("5. Delete Employee by ID");

Console.WriteLine("0. Exit");

}

void EmployeeInteraction()

{

int choice = 0;

do

{

PrintMenu();

Console.WriteLine("Please select an option");

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

switch (choice)

{

case 0:

Console.WriteLine("Bye.....");

break;

case 1:

AddEmployee();

break;

case 2:

PrintAllEmployees();

break;

case 3:

SearchAndPrintEmployee();

break;

case 4:

UpdateEmployeeNsmrById();

break;

case 5:

DeleteEmployeeByID();

break;

default:

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

break;

}

} while (choice != 0);

}

void AddEmployee()

{

if (employees[employees.Length - 1] != null)

{

Console.WriteLine("Sorry we have reached the maximum number of employees");

return;

}

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

{

if (employees[i] == null)

{

employees[i] = CreateEmployee(i);

}

}

}

void PrintAllEmployees()

{

if (employees[0] == null)

{

Console.WriteLine("No Employees available");

return;

}

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

{

if (employees[i] == null) continue;

PrintEmployee(employees[i]);

}

}

Employee CreateEmployee(int id)

{

Employee employee = new Employee();

employee.Id = 101 + id;

employee.BuildEmployeeFromConsole();

return employee;

}

void PrintEmployee(Employee employee)

{

Console.WriteLine("---------------------------");

employee.PrintEmployeeDetails();

Console.WriteLine("---------------------------");

}

int GetIdFromConsole()

{

int id = 0;

Console.WriteLine("Please enter the employee Id");

while (!int.TryParse(Console.ReadLine(), out id))

{

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

}

return id;

}

void SearchAndPrintEmployee()

{

Console.WriteLine("Print One employee");

int id = GetIdFromConsole();

Employee employee = SearchEmployeeById(id);

if (employee == null)

{

Console.WriteLine("No such Employee is present");

return;

}

PrintEmployee(employee);

}

void UpdateEmployeeNsmrById() {

int id = GetIdFromConsole();

Employee employee = SearchEmployeeById(id);

if (employee == null)

{

Console.WriteLine("No Such Employee is Present");

return;

}

Console.WriteLine($"Name - {employee.Name}");

Console.WriteLine("Please enter the updated name:");

employee.Name = Console.ReadLine() ?? string.Empty;

Console.WriteLine("Name has been updated") ;

}

void DeleteEmployeeByID()

{

int id = GetIdFromConsole();

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

{

if (id == employees[i].Id) {

employees[i] = null;

break;

}

}

}

Employee SearchEmployeeById(int id)

{

Employee employee = null;

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

{

if (employees[i] != null && employees[i].Id == id)

{

employee = employees[i];

break;

}

}

return employee;

}

static void Main(string[] args)

{

Program program = new Program();

program.EmployeeInteraction();

}

}

}

### EmployeeTrackerModelLibrary.Employee

namespace EmployeeTrackerModelLibrary

{

public class Employee

{

int age;

DateTime dob;

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public int Age

{

get

{

return age;

}

}

public DateTime DateOfBirth

{

get => dob;

set

{

dob = value;

age = ((DateTime.Today - dob).Days) / 365;

}

}

public double Salary { get; set; }

public Employee()

{

Id = 0;

Name = string.Empty;

Salary = 0.0;

DateOfBirth = new DateTime();

}

public Employee(int id, string name, DateTime dateOfBirth, double salary)

{

Id = id;

Name = name;

DateOfBirth = dateOfBirth;

Salary = salary;

}

public void BuildEmployeeFromConsole()

{

Console.WriteLine("Please enter the Name");

Name = Console.ReadLine() ?? String.Empty;

Console.WriteLine("Please enter the Date of birth");

DateOfBirth = Convert.ToDateTime(Console.ReadLine());

Console.WriteLine("Please enter the Basic Salary");

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

}

public void PrintEmployeeDetails()

{

Console.WriteLine("Employee Id : " + Id);

Console.WriteLine("Employee Name " + Name);

Console.WriteLine("Date of birth : " + DateOfBirth);

Console.WriteLine("Employee Age : " + Age);

Console.WriteLine("Employee Salary : Rs." + Salary);

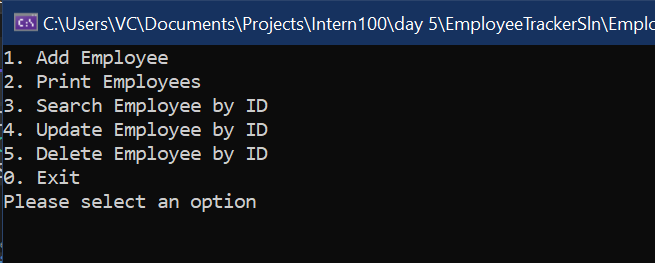
}

}

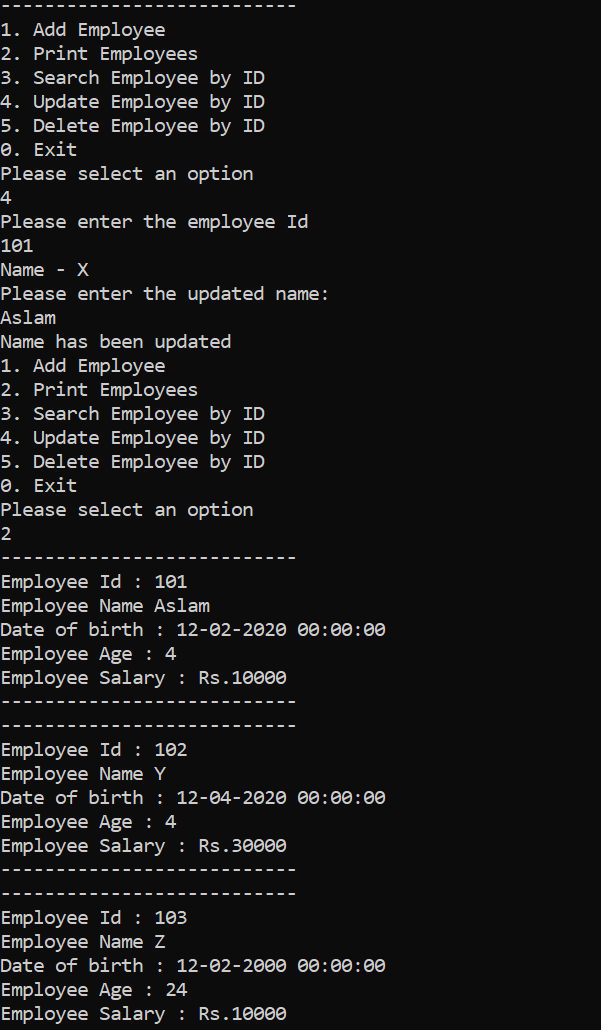
}

## Output

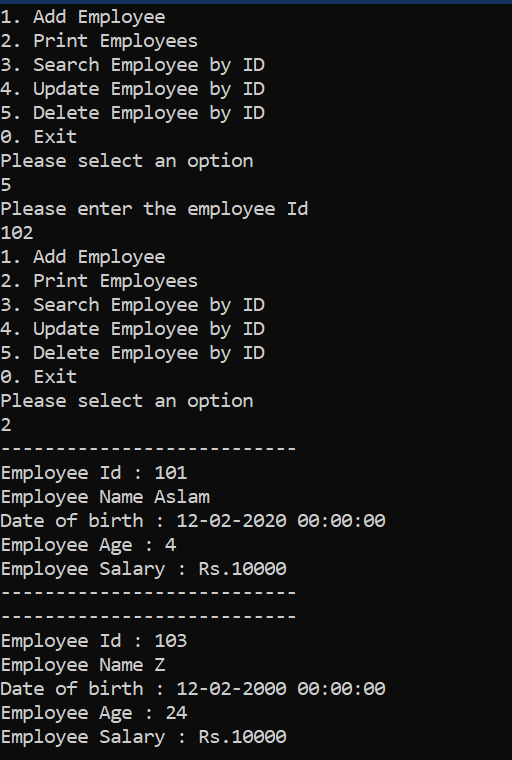
### Menu



### Updating Employee Name by Id



### Deleting Employee By ID



# Cow and Bulls Leetcode

## Code

public class Solution

{

public string GetHint(string secret, string guess)

{

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

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

int bullCount = 0;

int cowCount = 0;

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

{

if (secret[i] == guess[i])

{

bullCount++;

}

else

{

secretList.Add(secret[i]);

guessList.Add(guess[i]);

}

}

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

{

if (guessList.Contains(secretList[i]))

{

cowCount++;

guessList.Remove(secretList[i]);

}

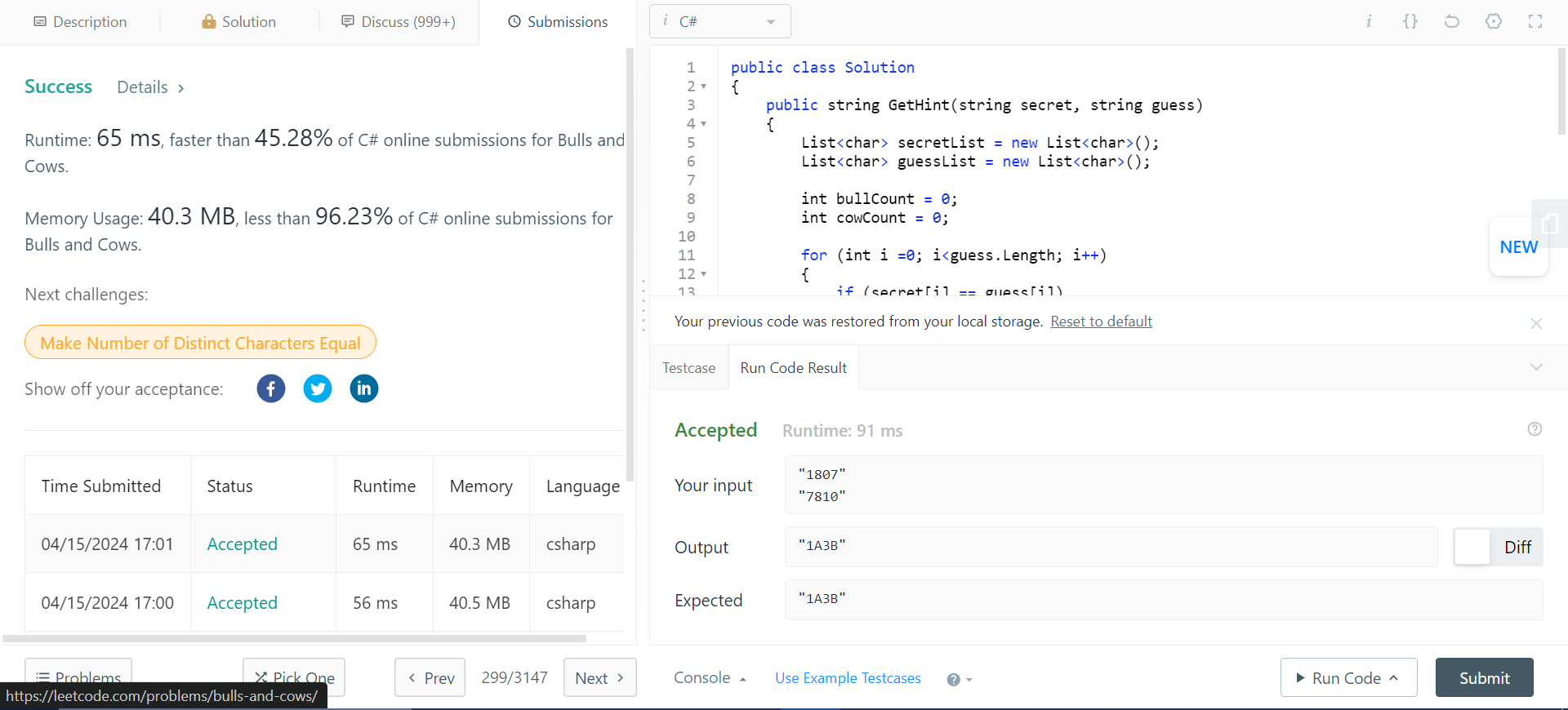
}

return string.Format("{0}A{1}B", bullCount, cowCount);

}

}

## Output



# Cows and Bulls

## Code

### CowsBulls.Program

using CowsBullsModelLib;

namespace CowsBulls

{

internal class Program

{

static void Main(string[] args)

{

CowsBullsGameService cowBullGame = new();

Player p1 = new("Andy");

Player p2 = new("Gandy");

Player[] players = [p1, p2];

int winner = cowBullGame.StartGame(players, "golf");

Console.WriteLine($"{players[winner].Name} wins");

}

}

}

### CowsBullsModelLib. CowsBullsGameService

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CowsBullsModelLib

{

public class CowsBullsGameService

{

private int Score { get; set; } = 0;

public int InitiateGame(string secret) {

int cows = 0; int bulls = 0;

HashSet<char> secretSet = secret.ToCharArray().ToHashSet();

while (cows != 4)

{

cows = 0; bulls = 0;

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

string guessed = Console.ReadLine() ?? String.Empty;

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

{

if (guessed[i] == secret[i])

{

cows++;

}

else if (secretSet.Contains(guessed[i]))

{

bulls++;

}

}

Console.WriteLine($"Cows: {cows}, Bulls: {bulls}");

Score++;

}

return Score;

}

public int InitiateGameV2(string secret) {

int steps = 0;

int cows = 0;

while(cows != 4)

{

string guess = Console.ReadLine() ?? String.Empty;

List<char> secretList = [];

List<char> guessList = [];

cows = 0;

int bulls = 0;

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

{

if (guess[i] == secret[i])

{

cows++;

}

else

{

secretList.Add(secret[i]);

guessList.Add(guess[i]);

}

}

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

if (guessList.Contains(secretList[i]))

{

bulls++;

guessList.Remove(secretList[i]);

}

}

Console.WriteLine($"Cows: {cows}, Bulls: {bulls}");

steps++;

}

return steps;

}

public int StartGame(Player[] players, string secret) {

foreach (Player p in players) {

Console.WriteLine($"{p.Name}'s turn!");

p.MyScore = InitiateGameV2(secret);

Console.Clear();

}

int winIdx = 0;

int highScore = int.MaxValue;

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

{

if (highScore > players[i].MyScore) {

highScore = players[i].MyScore;

winIdx = i;

}

}

return winIdx;

}

}

}

### CowsBullsModelLib.Player

namespace CowsBullsModelLib

{

public class Player

{

public string Name { get; set; }

public int MyScore { get; set; } = 0;

public Player(string name)

{

Name = name;

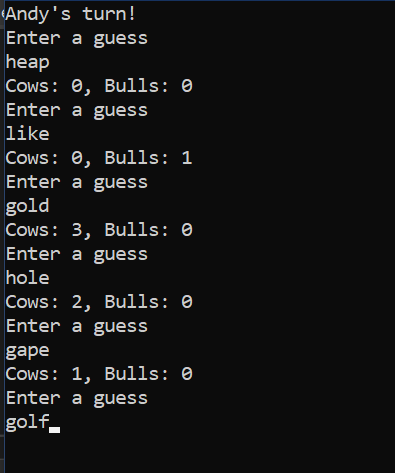
}

}

}

## Output

### Player 1 Turn



### Player 2 Turn

### 

### Result

