using System;

using System.Text.RegularExpressions;

using System.Linq;

//using System.Collections.Generic;

namespace Task02\_Emoji\_Detector

{

class Program

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string numbersPatern = @"\d";

Regex extractNumber = new Regex(numbersPatern);

MatchCollection numbersInText = extractNumber.Matches(text);

long coolThreshold = 1;

foreach (Match item in numbersInText)

{

coolThreshold \*= int.Parse(item.Value);

}

Console.WriteLine($"Cool threshold: {coolThreshold}");

string emojiesPatern = @"(\\*\\*|::)([A-Z][a-z]{2,})(\1)";

Regex extractEmojies = new Regex(emojiesPatern);

MatchCollection emojiesInText = extractEmojies.Matches(text);

Console.WriteLine($"{emojiesInText.Count} emojis found in the text. The cool ones are:");

foreach (Match item in emojiesInText)

{

int sumASCII = 0;

int count = 1;

foreach (char letter in item.Value)

{

if (1 < count && count < item.Value.Length - 2)

{

sumASCII += letter;

}

count++;

}

if (sumASCII >= coolThreshold)

{

Console.WriteLine(item);

}

}

/\*

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

foreach (Match item in emojiesInText)

{

int sumASCII = 0;

int count = 1;

foreach (char letter in item.Value)

{

if(1 < count && count < item.Value.Length - 2)

{

sumASCII += letter;

}

count++;

}

if(sumASCII >= coolThreshold)

{

coolEmojies.Add(item.Value);

}

}

if(coolEmojies.Count > 0)

{

Console.WriteLine($"{emojiesInText.Count} emojis found in the text. The cool ones are:");

foreach (var item in coolEmojies)

{

Console.WriteLine(item);

}

}

\*/

}

}

}

using System;

using System.Text.RegularExpressions;

using System.Linq;

//using System.Collections.Generic;

namespace Task02\_Emoji\_Detector

{

class Program

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string numbersPatern = @"\d";

Regex extractNumber = new Regex(numbersPatern);

MatchCollection numbersInText = extractNumber.Matches(text);

long coolThreshold = 1;

foreach (Match item in numbersInText)

{

coolThreshold \*= int.Parse(item.Value);

}

Console.WriteLine($"Cool threshold: {coolThreshold}");

string emojiesPatern = @"(\\*\\*|::)([A-Z][a-z]{2,})(\1)";

Regex extractEmojies = new Regex(emojiesPatern);

MatchCollection emojiesInText = extractEmojies.Matches(text);

Console.WriteLine($"{emojiesInText.Count} emojis found in the text. The cool ones are:");

foreach (Match item in emojiesInText)

{

int sumASCII = 0;

int count = 1;

foreach (char letter in item.Value)

{

if (1 < count && count < item.Value.Length - 2)

{

sumASCII += letter;

}

count++;

}

if (sumASCII >= coolThreshold)

{

Console.WriteLine(item);

}

}

/\*

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

foreach (Match item in emojiesInText)

{

int sumASCII = 0;

int count = 1;

foreach (char letter in item.Value)

{

if(1 < count && count < item.Value.Length - 2)

{

sumASCII += letter;

}

count++;

}

if(sumASCII >= coolThreshold)

{

coolEmojies.Add(item.Value);

}

}

if(coolEmojies.Count > 0)

{

Console.WriteLine($"{emojiesInText.Count} emojis found in the text. The cool ones are:");

foreach (var item in coolEmojies)

{

Console.WriteLine(item);

}

}

\*/

}

}

}

using System;

using System.Linq;

using System.Collections.Generic;

namespace Task03\_P\_rates

{

class Program

{

static void Main(string[] args)

{

Dictionary<string, List<int>> cityTargets = new Dictionary<string, List<int>>();

while (true)

{

string[] cityDatas = Console.ReadLine().Split("||", StringSplitOptions.RemoveEmptyEntries);

if(cityDatas[0].ToUpper() == "SAIL")

{

break;

}

string name = cityDatas[0];

int people = int.Parse(cityDatas[1]);

int gold = int.Parse(cityDatas[2]);

if(cityTargets.ContainsKey(name))

{

cityTargets[name][0] += people;

cityTargets[name][1] += gold;

}

else

{

cityTargets.Add(name, new List<int> { people, gold });

}

}

/\*

foreach (var item in cityTargets)

{

Console.WriteLine($"{item.Key} : {item.Value[0]} : {item.Value[1]}");

}

\*/

while (true)

{

string[] action = Console.ReadLine().Split("=>", StringSplitOptions.RemoveEmptyEntries);

if(action[0].ToUpper() == "END")

{

break;

}

string typeAction = action[0];

switch (typeAction.ToUpper())

{

case "PLUNDER":

string name = action[1];

int people = int.Parse(action[2]);

int gold = int.Parse(action[3]);

cityTargets[name][0] -= people;

cityTargets[name][1] -= gold;

Console.WriteLine($"{name} plundered! {gold} gold stolen, {people} citizens killed.");

if (cityTargets[name][0] <= 0 || cityTargets[name][1] <= 0)

{

cityTargets.Remove(name);

Console.WriteLine($"{name} has been wiped off the map!");

}

break;

case "PROSPER":

name = action[1];

gold = int.Parse(action[2]);

if(gold < 0)

{

Console.WriteLine($"Gold added cannot be a negative number!");

}

else

{

cityTargets[name][1] += gold;

Console.WriteLine($"{gold} gold added to the city treasury. {name} now has {cityTargets[name][1]} gold.");

}

break;

default:

break;

}

}

/\*

foreach (var item in cityTargets)

{

Console.WriteLine($"{item.Key} : {item.Value[0]} : {item.Value[1]}");

}

\*/

if(cityTargets.Count > 0)

{

Console.WriteLine($"Ahoy, Captain! There are {cityTargets.Count} wealthy settlements to go to:");

foreach (var item in cityTargets.OrderByDescending(x => x.Value[1]).ThenBy(x => x.Key))

{

Console.WriteLine($"{item.Key} -> Population: {item.Value[0]} citizens, Gold: {item.Value[1]} kg");

}

}

else

{

Console.WriteLine("Ahoy, Captain! All targets have been plundered and destroyed!");

}

}

}

}