using System;

using System.Text;

namespace Task01\_Password\_Reset

{

class Program

{

static void Main(string[] args)

{

string rawPassword = Console.ReadLine();

while (true)

{

string[] comand = Console.ReadLine().Split(' ', StringSplitOptions.RemoveEmptyEntries);

if(comand[0].ToUpper() == "DONE")

{

break;

}

string typeComand = comand[0].ToUpper();

switch (typeComand)

{

case "TAKEODD":

string tempHelpPassword = string.Empty;

for (int i = 1; i < rawPassword.Length; i++)

{

if(i % 2 != 0)

{

tempHelpPassword += rawPassword[i];

}

}

rawPassword = tempHelpPassword;

Console.WriteLine(rawPassword);

break;

case "CUT":

int startIndex = int.Parse(comand[1]);

int lenght = int.Parse(comand[2]);

rawPassword = rawPassword.Remove(startIndex, lenght);

Console.WriteLine(rawPassword);

break;

case "SUBSTITUTE":

string cutSubstring = comand[1];

string pasteSubstring = comand[2];

if(rawPassword.Contains(cutSubstring))

{

rawPassword = rawPassword.Replace(cutSubstring, pasteSubstring);

Console.WriteLine(rawPassword);

}

else

{

Console.WriteLine("Nothing to replace!");

}

break;

default:

break;

}

}

Console.WriteLine($"Your password is: {rawPassword}");

}

}

}

using System;

using System.Text.RegularExpressions;

namespace Task02\_Fancy\_Barcodes

{

class Program

{

static void Main(string[] args)

{

string patern = @"(@#+)([A-Z][\d|[a-zA-Z)]{4,}[A-Z])(@#+)"; // @"(@#+)([A-Z][\d|[A-z)]{4,}[A-Z])(@#+)"; //@"(@#+)([A-Z][[[:alnum:]]{4,}[A-Z])(@#+)";

Regex validBarCode = new Regex(patern);

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

for (int i = 1; i <= productsNumber; i++)

{

string productGroup = string.Empty;

string nextProduct = Console.ReadLine();

MatchCollection productBarCode = validBarCode.Matches(nextProduct);

if (productBarCode.Count > 0)

{

string helpName = productBarCode[0].ToString();

//string helpName = productBarCode.ToString();

string digitPatern = @"\d";

Regex extractDigits = new Regex(digitPatern);

MatchCollection digits = extractDigits.Matches(helpName);

foreach (Match digit in digits)

{

productGroup += digit;

}

if(productGroup.Length < 1)

{

productGroup = "00";

}

Console.WriteLine($"Product group: {productGroup}");

}

else

{

Console.WriteLine("Invalid barcode");

//break;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

namespace Task03\_Heroes\_of\_Code\_and\_Logic\_VII

{

class Program

{

static void Main(string[] args)

{

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

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

for (int i = 1; i <= numberOfHeroes; i++)

{

string[] nextHeroDatas = Console.ReadLine().Split(' ', StringSplitOptions.RemoveEmptyEntries);

string name = nextHeroDatas[0];

int hitPoint = int.Parse(nextHeroDatas[1]);

int manaPoint = int.Parse(nextHeroDatas[2]);

heroes.Add(name, new List<int> { hitPoint, manaPoint });

}

/\*

foreach (var item in heroes)

{

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

}

\*/

while (true)

{

string[] comand = Console.ReadLine().Split(" - ", StringSplitOptions.RemoveEmptyEntries);

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

{

break;

}

string typeAction = comand[0];

switch (typeAction.ToUpper())

{

case "CASTSPELL":

string name = comand[1];

int manaPointNeeded = int.Parse(comand[2]);

string spellName = comand[3];

if(heroes[name][1] >= manaPointNeeded)

{

heroes[name][1] -= manaPointNeeded;

Console.WriteLine($"{name} has successfully cast {spellName} and now has {heroes[name][1]} MP!");

}

else

{

Console.WriteLine($"{name} does not have enough MP to cast {spellName}!");

}

break;

case "TAKEDAMAGE":

name = comand[1];

int damage = int.Parse(comand[2]);

string attacker = comand[3];

if(heroes.ContainsKey(name))

{

heroes[name][0] -= damage;

if(heroes[name][0] <= 0)

{

heroes.Remove(name);

Console.WriteLine($"{name} has been killed by {attacker}!");

}

else

{

Console.WriteLine($"{name} was hit for {damage} HP by {attacker} and now has {heroes[name][0]} HP left!");

}

}

break;

case "RECHARGE":

name = comand[1];

int amount = int.Parse(comand[2]);

if (heroes[name][1] + amount > 200)

{

amount = 200 - heroes[name][1];

}

heroes[name][1] += amount;

Console.WriteLine($"{name} recharged for {amount} MP!");

break;

case "HEAL":

name = comand[1];

amount = int.Parse(comand[2]);

if (heroes[name][0] + amount > 100)

{

amount = 100 - heroes[name][0];

}

heroes[name][0] += amount;

Console.WriteLine($"{name} healed for {amount} HP!");

break;

default:

break;

}

}

foreach (var item in heroes.OrderByDescending(x => x.Value[0]).ThenBy(x => x.Key))

{

Console.WriteLine(item.Key);

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

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

}

}

}

}