KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:

IFF-7/14 gr. studentas

Eigijus Kiudys

2020 m. vasario 24 d.

Priėmė:

lekt. Evaldas Guogis

lekt. Fyleris Tautvydas

KAUNAS 2020

TURINYS

1. Python (L1) 3
1.1. Darbo užduotis 3
1.2. Programos tekstas 4
1.3. Pradiniai duomenys ir rezultatai 6

1. Python (L1)

1.1. Darbo užduotis

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=3&page=show_problem&problem=75

A large company wishes to monitor the cost of phone calls made by its personnel. To achieve this the PABX logs, for each call, the number called (a string of up to 15 digits) and the duration in minutes. Write a program to process this data and produce a report specifying each call and its cost, based on standard Telecom charges.

International (IDD) numbers start with two zeroes (00) followed by a country code (1–3 digits) followed by a subscriber's number (4–10 digits). National (STD) calls start with one zero (0) followed by an area code (1–5 digits) followed by the subscriber's number (4–7 digits). The price of a call is determined by its destination and its duration. Local calls start with any digit other than 0 and are free.

Input

Input will be in two parts. The first part will be a table of IDD and STD codes, localities and prices as follows:

Code △ *Locality name*$*price in cents per minute*

where △ represents a space. Locality names are 25 characters or less. This section is terminated by a line containing 6 zeroes (`000000`).

The second part contains the log and will consist of a series of lines, one for each call, containing the number dialled and the duration. The file will be terminated a line containing a single `#`. The numbers will not necessarily be tabulated, although there will be at least one space between them. Telephone numbers will not be ambiguous.

Output

Output will consist of the called number, the country or area called, the subscriber's number, the duration, the cost per minute and the total cost of the call, as shown below. Local calls are costed at zero. If the number has an invalid code, list the area as '`Unknown`' and the cost as −1.00.

Note: The first line of the Sample Output below in not a part of the output, but only to show the exact tabulation format it must follow.

Sample Input

```
088925 Broadwood$81
03 Arrowtown$38
0061 Australia$140
000000
031526         22
0061853279   3
0889256287213    122
779760 1
002832769 5
#
```

Sample Output

```
1               17                               51   56    62     69
031526          Arrowtown                      1526   22  0.38   8.36
0061853279      Australia                    853279    3  1.40   4.20
0889256287213   Broadwood                   6287213  122  0.81  98.82
779760          Local                        779760    1  0.00   0.00
002832769       Unknown                                5        -1.00
```

1.2. Programos tekstas

```
class TelephoneInfo:
    def __init__(self, code, name, price):
        self.code = code
        self.name = name
        self.price = price

    def priceSeconds(self):
        return round(self.price * 0.1, 2)

class TelephoneCalls:
    def __init__(self, number, time):
        self.number = number
        self.time = time

class Data (TelephoneInfo, TelephoneCalls):
    def __init__(self, number, code, name, price, time):
        self.number = number
        self.code = code
        self.name = name
        self.price = price
        self.time = time

    def calcuPrice(self):
        return float(self.price) * 0.1 * float(self.time)

    def __str__(self):
        if self.code != -1:
            return "{0:15} {1:16} {2:8} {3:6} {4:6} {5:.2f}\n".format(self.number,
self.name, self.code, self.time, str(self.priceSeconds()), self.calcuPrice())
        else:
            return "{0:15} {1:16} {2:8} {3:6} {4:6} {5:.2f}\n".format(self.number,
self.name, "", self.time, "", float(self.price))


class Main:

    def __init__(self, read, write):
        self.readFromFile = read
        self.writeToFile = write

    def dataRead(self):
        data = []
        file = open(self.readFromFile, "r")
        for dataFromFile in file:
            data.append(dataFromFile)
        file.close()
        return data

    def splitLine(self,dataFromFile):
        split = []
        listForInfo = []
        listForCalls = []
        info = False
        for i in dataFromFile:

            index = 0
            if len(split) == 2 and info:
                split = []
            elif not info:
                split = []

            for string in i:
                if string == " ":
                    split.append(i[0:index])
                    split.append(i[index + 1: len(i)])
                    break
                index += 1
            if index == len(i):
                split.append(i.strip())

            split[0] = split[0].strip()
            if len(split) > 1 and not bool(info):
                split[1] = split[1].strip()
                data = split[1].split('$')
                telephoneInfo = TelephoneInfo(split[0], data[0], float(data[1]) * 0.1)
                listForInfo.append(telephoneInfo)
            elif len(split) == 1 and split[0] == "000000" and not bool(info):
                split = []
                info = True
            elif len(split) > 1 and bool(info):
                split[1] = split[1].strip()
                calls = TelephoneCalls(split[0], split[1])
                listForCalls.append(calls)
        return listForInfo, listForCalls

    def calculatePrice(self, listOfCall, listOfInfo):
        calculatedData = []
        for calls in listOfCall:
            state = False
            for info in listOfInfo:
                if calls.number[0:len(info.code)] == info.code:
                    data = Data(calls.number,
calls.number[len(info.code):len(calls.number)], info.name, info.price,
                                calls.time)
                    calculatedData.append(data)
                    state = True
                    break
                elif calls.number[0] != "0":
                    tempInfo = TelephoneInfo(calls.number, "Local", 0)
                    data = Data(calls.number, tempInfo.code, tempInfo.name,
tempInfo.price, calls.time)
                    calculatedData.append(data)
                    state = True
                    break
            if not bool(state):
                data = Data(calls.number, -1, "Unknown", -1, calls.time)
                calculatedData.append(data)
        return calculatedData

    def saveData(self, calculatedData):
        fSave = open(self.writeToFile, "w+")
        for data in calculatedData:
            fSave.write(str(data))
        fSave.close()

    def run(self):
        fileData = self.dataRead()
        infoList, callsList = self.splitLine(fileData)
        dataList = self.calculatePrice(callsList, infoList)
        self.saveData(dataList)

main = Main("test.txt", "data.txt")
main.run()
```

1.3. Pradiniai duomenys ir rezultatai

Pradiniai duomenys	Rezultatai
088925 Broadwood$81 03 Arrowtown$38 0061 Australia$140 000000 031526 22 0061853279 3 0889256287213 122 779760 1 002832769 5 #	031526 Arrowtown 1526 22 0.38 8.36 0061853279 Australia 853279 3 1.4 4.20 0889256287213 Broadwood 6287213 122 0.81 98.82 779760 Local 779760 1 0.0 0.00 002832769 Unknown 5 -1.00