2018-2019 秋季学期程序设计 I 期中测试

中国人民大学　信息学院

注意事项

1. 本次测试的时间为 180 分钟；编程结果采用机器自动评测。
2. 共有 5 题，第 1、2 题，25 分；第 3 题，20 分；第 4、5 题，15 分。
3. 提交到在线评测系统中的程序均采用标准输入和标准输出（键盘输入和屏幕输出）。
4. 程序设计语言选用 C 或 C++。
5. 所有题目的时间限制均为 1s。

1. 分段函数

分段函数

已知有如下分段离散函数f (x)，其中自变量x为[-1000, 1000]之间的实数。请编写一个程序计算函数f(f(x)) 的值。注意，当函数值有小数时四舍五入保留2位小数。

$$f(x)=\begin{cases}|x-1|-2 & (|x|\le 1);\\ \dfrac{1}{1+x^2} & (|x|>1);\end{cases}$$

输入格式

输入一个实数x。

输出格式

输出f(f(x))函数值，四舍五入保留2位小数。

评测样例

样例 1 ≫

```
0.5
```

≪ 样例 1

```
0.31
```

样例 2 ≫

```
-3
```

≪ 样例 2

```
-1.10
```

数据规模和约定

浮点数计算过程中请使用double类型。

参考代码

```c
#include <stdio.h>
#include <math.h>
double f(double x)
{
    if(x >= -1 && x <= 1)
    {
        return fabs(x - 1) - 2;
    }
    else return 1.0 / (1 + x * x);
}
int main(int argc, char *argv[]) {
    double x;
    scanf("%lf", &x);
    double ans = f(f(x));
    printf("%.2lf", ans);
    return 0;
}
```

2. 外侧元素求和

外侧元素求和

求一个n×m矩阵 [a_ij] 的所有靠外侧的元素值之和。外侧元素是指，矩阵四周外侧的元素，即首行和末行，及首列和末列的所有元素（注意每个元素只算一遍，不要重复计算）。取值范围：1≤n, m≤100；矩阵元素为整数 –99,999≤a_ij≤99,999。

输入格式

输入分为n+1行。

第一行为n和m，n为矩阵行数，m为列数，之间由空格分隔；第二行至n+1行为矩阵的元素，按行排列矩阵的元素，数据之间由空格分隔，行以回车结束。

输出格式

输出一个整数，为求和结果。

评测样例

- 样例 1 ≫

```
3 4
3 8 9 10
2 5 -3 5
7 0 -1 4
```

- ≪ 样例 1

```
47
```

- 样例 2 ≫

```
1 1
100
```

- ≪ 样例 2

```
100
```

参考代码

```c
#include <stdio.h>
int main(int argc, char *argv[]) {
    int n, m;
    scanf("%d%d", &n, &m);
    int Sum = 0;
    int TmpIn;
    for(int i = 0; i < n; i++)
    {
```

```
9           for(int j = 0; j < m; j++)
10          {
11              scanf("%d", &TmpIn);
12              if(i == 0 || i == n - 1)
13              {
14                  Sum += TmpIn;
15              }
16              else if(j == 0 || j == m - 1)
17              {
18                  Sum += TmpIn;
19              }
20          }
21      }
22      printf("%d", Sum);
23      return 0;
24  }
```

3. 共享单车停放

共享单车停放

城市里共享单车越来越多，给人们带来方便的同时，单车的乱停放问题也给管理者与城市交通带来了一定困扰。共享单车定点停放势在必行，于是对共享单车App产生了新的功能需求：为用车人推荐一个目的地附近的停车区，保证用车人到达后有单车停车位且步行到目的地的距离最短。
现在，小明同学要从人民大学东门骑车到国家图书馆，为了给他推荐一个符合要求的停车区，手机App需得到如下图表所示的信息。表中空余车位变化速度为正数时，表示被骑走的车比停入的车多，直到车位全空（不必在意全空时车位具体有多少）；反之空余车位变化速度为负数时，停入的车比骑走的车多，直到全满，剩余车位数为0。这里有些数据（如车位变化速度和预估的骑行时间）来源于物联网和大数据技术的监测和分析。为了简化计算，假定剩余车位变化速度在小明骑行期间不变。
根据这些信息，手机App可引导小明直接骑到4号区所在的位置。
现根据App上述需求，当小明要骑车去别的地方时，为其推荐一个停车区的位置（编号，从1开始）。

国图阅览室 1000 米范围内可停车区的数量：	5				
停车区编号：	停车区 1	停车区 2	停车区 3	停车区 4	停车区 5
国图阅览室到各停车区的距离：	80	90	100	150	160
各停车区当前的剩余车位数：	20	5	10	0	15
停车区空余车位变化速度（个/每 10 分钟）：	-7	-3	-5	2	-5
从人大东门骑到停车区的预估时间（分钟）：	30	20	20	20	20

输入格式

共5行，第1行，1个整数n，表示单车停放区数量。

之后的4行，每行n个整数。其含义如上表所示，分别为目的地点到各单车区的距离，各停车区的剩余车位数，空余车位变化速度，以及小明骑到停车区的预估时间。各行数字的次序按停车区一一对应，数字之间均为空格分隔。

输出格式

1行仅1个整数，表示推荐的停车区的编号。如果找不到合适停车区，输出0。

评测样例

- 样例 >>

```
5
80 90 100 150 160
20 5 10 0 15
-7 -3 -5 2 -5
30 20 20 20 20
```

- << 样例

```
4
```

数据规模和约定

任意目的地周边停靠点最多不超过50个，且距离各不相同；更新剩余车位时，使用round库函数对变化数进行四舍五入取整。

参考代码

```c
#include <stdio.h>
#include <math.h>
typedef struct
{
    int Num;
    int Dis;
    int Rem;
    int Chg;    //Per 10mins
    int Time;
}Park;
int main(int argc, char *argv[]) {
    Park Src[50];
    int Total;
    scanf("%d", &Total);
    for(int j = 0; j < Total; j++)
        Src[j].Num = j + 1;
    for(int j = 0; j < Total; j++)
        scanf("%d", &Src[j].Dis);
    for(int j = 0; j < Total; j++)
        scanf("%d", &Src[j].Rem);
    for(int j = 0; j < Total; j++)
        scanf("%d", &Src[j].Chg);
    for(int j = 0; j < Total; j++)
        scanf("%d", &Src[j].Time);

    for(int i = 0; i < Total; i++)
        Src[i].Rem = Src[i].Rem + round(Src[i].Chg * Src[i].Time / 10.0);
    //Sort
    for(int i = 0; i < Total - 1; i++)
    {
        int ptr = i;
        for(int j = i + 1; j < Total; j++)
        {
            if(Src[j].Dis < Src[ptr].Dis)
                ptr = j;
        }
```

```
37          if(ptr != i)
38          {
39              Park Tmp;
40              Tmp = Src[i];
41              Src[i] = Src[ptr];
42              Src[ptr] = Tmp;
43          }
44      }
45      for(int i = 0; i < Total; i++)
46      {
47          if(Src[i].Rem > 0)
48          {
49              printf("%d", Src[i].Num);
50              return 0;
51          }
52      }
53      printf("0");
54      return 0;
55  }
```

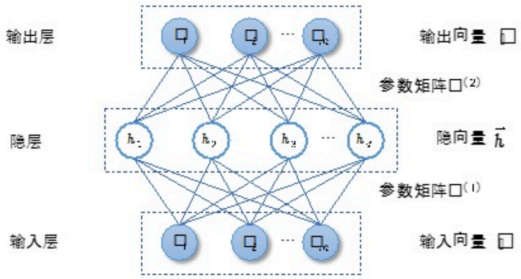
4. 深度神经网络

深度神经网络

一种最基本的深度神经网络——多层感知器模型（Multi-Layer Perceptron, MLP），见下图描述。

【问题描述】

美国谷歌公司开发的 Alpha Go 围棋程序接连战胜李世石、柯洁等人类顶尖棋手，其成功秘诀就在于使用了深度神经网络。现请你编写程序实现深度神经网络的一种最基本模型——多层感知器模型（Multi-Layer Perceptron, MLP）。



一个最基本的多层感知器模型（单隐层）如上图所示，它通过一系列非线性变换将输入层的列向量$\vec{x}=(x_1,x_2,\ldots,x_m)^T$，变换成隐层中的向量$\vec{h}=(h_1,h_2,\ldots,h_d)^T$，进而变换成输出层的列向量$\vec{o}=(o_1,o_2,\ldots,o_n)^T$。注：图中的虚线方框表示一个向量，内部的圆圈表示向量的数值，连线表示向量的变换。

具体来讲，将输入向量$\vec{x}$变换为隐向量 $\vec{h}$，计算公式的矩阵表达形式为：

$$\vec{h}=s(W\vec{x})$$

其中 W 为$d\times m$维的参数矩阵，$W\vec{x}$表示矩阵 W 与列向量$\vec{x}$的乘积。s表示向量的函数变换，该函数对向量中的每一个维度数值a进行计算变换，常用的一种变换是：$s(a)=1/(1+e^{-a})$。所有维度的计算结果就构成所需的向量$\vec{h}$。例如：向量$(0,0)^T$经过s函数计算后的结果为$(0.5,0.5)^T$。

与此类似，将向量$\vec{h}$变换为输出层的列向量$\vec{o}$，计算公式的矩阵表达形式为：

$$\vec{o}=s(V\vec{h})$$

其中 V 为n × d维的参数矩阵，$V\vec{h}$表示矩阵 V 与列向量$\vec{h}$的乘积。s同样是向量上的函数变换，含意同上。

现给你提供输入向量$\vec{x}$、参数矩阵 W 和 V，请你计算输出向量$\vec{o}$。注意，输出结果保留两位小数。

【输入格式】

第 1 行：3 个整数，分别表示输入向量$\vec{x}$的维度m，隐向量 $\vec{h}$ 的维度d，以及输出向量 $\vec{o}$ 的维度n；**第 2 行**：输入向量 $\vec{x}$，共m个整数；**第 3 行**：参数矩阵 W 中的 d × m 个整数，其次序与该矩阵中数字按行（首尾相连）次序一致；**第 4 行**：参数矩阵 V 中的 n × d 个整数，其排列方式同前。数据之间都以空格分隔。

【输出格式】

输出一行，为计算结果 $\vec{o}$ 向量，每个维度数值保留两位小数，之间以空格分隔

【数据规模说明】

1. 题目中的向量维度不超过 16、矩阵行与列的维度均不超过 16；
2. 向量和矩阵中的数字均在 int 类型表数范围内。

【提示信息】

1. 数值计算请使用 double 数据类型；
2. 指数函数请使用 math.h 头文件中的 double exp(double x)函数。
3. 矩阵乘以列向量的公式：以前面的$W\vec{x}$为例，结果为d维的列向量，可以表示为$(\sum_{i=1}^{m}w_{1i}x_i,\sum_{i=1}^{m}w_{2i}x_i,\ldots,\sum_{i=1}^{m}w_{di}x_i)^T$

输入格式

见上图

输出格式

见上图

参考代码

```
#include <stdio.h>
#include <math.h>
//做矩阵与向量的乘法，结果存在向量数组里
int Mul(double MatIn[][16], int Row, int Col, double VectIn[])
{
    double TmpVec[16] = {0};
    for(int j = 0; j < Row; j++)
        for(int i = 0; i < Col; i++)
            TmpVec[j] += MatIn[j][i] * VectIn[i];
    for(int j = 0; j < Row; j++)
        VectIn[j] = TmpVec[j];
    return 0;
}
int Funcs(double Vec[], int Col)
{
    for(int i = 0; i < Col; i++)
        Vec[i] = 1.0 / (1 + exp(-Vec[i]));
    return 0;
}
int main(int argc, char *argv[]) {
    int m, d, n;
    scanf("%d%d%d", &m, &d, &n);
    double Matrix[16][16] = {0};
    double Vector[16] = {0};
    //输入向量x
    for(int i = 0; i < m; i++)
        scanf("%lf", &Vector[i]);
    //输入矩阵W
    for(int i = 0; i < d * m; i++)
        scanf("%lf", &Matrix[i / m][i % m]);
    Mul(Matrix, d, m, Vector);
    Funcs(Vector, d);
    //输入矩阵V
    for(int i = 0; i < n * d; i++)
        scanf("%lf", &Matrix[i / d][i % d]);
    Mul(Matrix, n, d, Vector);
```

```
37      Funcs(Vector, n);
38      for(int i = 0; i < n; i++)
39          printf("%.2lf ", Vector[i]);
40      return 0;
41  }
```

5. 班会时间

班会时间

由于班上同学所选课程和上课时段各不相同。为了找到一个合适的时间开班会，班长小A收集了班上所有同学的课表。
请你编写程序，帮助小A处理所有同学的课表，并得到开班会的合适时间段，时间段按照中国人民大学的课表划分（每个时间段的表示法如下表所示）。要求程序根据输入的所有课表，统计出每个时间段有课的学生人数，并将时间段按照上课人数从小到大排序，如果某两个时间段的上课人数一样，则按照时间段的先后顺序排序（时间在前的排在前面），输出有课人数前k（1≤k≤49）少的时间段。

节次	时间	周一	周二	周三	周四	周五	周六	周日
1	8:00-9:30	1.1	2.1	3.1	4.1	5.1	6.1	7.1
2	10:00-11:30	1.2	2.2	3.2	4.2	5.2	6.2	7.2
3	12:00-13:30	1.3	2.3	3.3	4.3	5.3	6.3	7.3
4	14:00-15:30	1.4	2.4	3.4	4.4	5.4	6.4	7.4
5	16:00-17:30	1.5	2.5	3.5	4.5	5.5	6.5	7.5
6	18:00-19:30	1.6	2.6	3.6	4.6	5.6	6.6	7.6
7	19:40-21:10	1.7	2.7	3.7	4.7	5.7	6.7	7.7

输入格式

第一行包含2个正整数n和k，分别表示班级的人数n和题目中要求的k。
接下来的n行，每行表示一个学生的课表，具体格式如下：
2017101234 10 2.1 1.2 4.3 2.4 2.5 5.1 5.2 1.6 4.2 4.4 。其中第1个十位整数m表示这个学生的学号，第2个正整数p表示这个学生有p个时间段有课，之后跟着p个小数表示有课的时间段，例如2.1表示周二第1节有课。每2个数字之间用一个空格隔开。

输出格式

输出k行，每行包含一个小数和一个整数，分别表示时间段和该时间段有课的学生人数，按照有课人数从小到大的顺序输出前k个，如果某两个时间段的上课人数一样，则时间在前面的先输出。

评测样例

- 样例 >>

```
3 5
2017101000 10 2.1 1.2 4.3 2.4 2.5 5.1 5.2 1.6 4.2 4.4
2017101001 9 2.1 2.2 2.5 3.2 1.1 1.3 4.3 4.4 5.2
2017101002 10 1.4 1.5 2.1 2.2 3.4 3.4 4.1 4.6 5.4 5.5
```

- << 样例

```
1.7 0
2.3 0
2.6 0
2.7 0
3.1 0
```

数据规模和约定

【数据规模说明】
40%的数据，k = 1;
100%的数据，1 ⩽ n ⩽ 1000, 1 ⩽ k ⩽ 49, 1 ⩽ p ⩽ 49，输入数据一个学生在同一个时间段至多只有1门课。

参考代码

```
#include <stdio.h>
int main()
{
    int Table[7][7] = {0};
    int Flag[7][7] = {0};
    int n, k;
    scanf("%d%d", &n, &k);
    int Num, Class;
    int Day, Period;
    for(int i = 0; i < n; i++)
    {
        scanf("%d%d", &Num, &Class);
        for(int j = 0; j < Class; j++)
        {
            scanf("%d.%d", &Day, &Period);
            Table[Day - 1][Period - 1]++;
        }
    }
    int MinI, MinJ;
    for(int p = 0; p < k; p++)
    {
        for(int i = 0; i < 49; i++)
        {
            if(Flag[i / 7][i % 7] == 0)
            {
                MinI = i / 7, MinJ = i % 7;
                break;
            }
        }
        for(int i = 0; i < 7; i++)
        {
            for(int j = 0; j < 7; j++)
            {
                if(Table[i][j] < Table[MinI][MinJ] && Flag[i][j] == 0)
                {
                    MinI = i;
```

```
                    MinJ = j;
                }
            }
        }
        if(Flag[MinI][MinJ] == 0)
        {
            printf("%d.%d %d\n", MinI + 1, MinJ + 1, Table[MinI][MinJ]);
            Flag[MinI][MinJ] = 1;
        }
    }
    return 0;
}
```