|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **清华大学本科生考试试题专用纸**  考试课程 计算机语言与程序设计 （ A 卷） 2019 年 12 月 15 日  姓名 请填写姓名 班级 请填写班级 学号 请填写学号   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 大题 | 题号 | 一 | 二 | 三 | 四 | 总分 | | 分数 | **30分** | **30分** | **30分** | **10分** | | 成绩 | |  |  |  |  |  |   **试卷提交说明**   1. **在考卷上填写你的姓名、学号；** 2. **所有程序，都只能使用C89/C99标准包含的头文件；** 3. **按要求编程，调试运行，将每道题的结果考屏，粘贴在考卷对应的题后；** 4. **交卷时，提交一个压缩文件包，包括：**    1. **文件包名“学号\_姓名.zip/.rar”，如2019123456\_张三.zip/.rar**    2. **ZIP/RAR压缩文件包内容**       * + - **每道题的c语言源文件，文件名为“学号\_题号.c”,如2019123456\_1.c；**           - **考卷，文件名为“学号\_考卷.docx”**           - **例如：2019123456.zip应包括2019123456\_1.c、2019123456\_2.c、2019123456\_3.c、2019123456\_4.c、2019123456\_考卷.docx，共5个文件。** 5. **字符串处理（30分）**   一种字符串加密技术中，明文中的所有字母都在字母表上向后按照一个固定数目（称为密钥）进行偏移后被替换成密文。例如，当密钥是3的时候，所有的字母a/A将被替换成d/D，b/B变成e/E，以此类推，x/X变成a/A，y/Y变成b/B，z/Z变成c/C。已知明文中字母e/E出现次数最多，因此通过分析密文字母出现的次数，可以估计出正确的密钥并解密出明文。编制一个程序，实现密文输入、密钥估计和明文输出，程序结构如下：   1. **函数unsigned int Input(char\* Buffer)（3分）**   从键盘输入密文字符串，存储在Buffer参数所指向的内存空间中（长度不超过1024），并通过函数返回值返回所输入的字符串长度。   1. **函数unsigned int InputKey()（2分）**   从键盘输入一个1~25的整数作为密钥，并通过函数返回值返回给主函数。   1. **函数void Cipher(char\* Buffer, unsigned int Len, int Key)（7分）**   通过Buffer、Len和Key参数分别接收主函数提供的明文字符串、密明文实际长度和密钥，将加密后的密文存放在Buffer参数所指向的内存空间中。只加密英文字符，空格、标点符号等其它字符都不加密。   1. **函数int GetKey(char\* Buffer, unsigned int Len) （6分）**   通过参数接收主函数提供的密文字符串和密文实际长度，对密文中每种英文字母统计出现次数（同一个英文字母的大小写形式算一种，如a和A出现的次数统计在一起），找到出现次数最多的密文字母，认为其就是明文e/E经过密钥位移后的字母（不考虑多个字母出现次数相同且都最多的情况）。通过函数返回值返回找到的密钥Key。   1. **函数void Decipher(char\* Buffer, unsigned int Len, int Key)（7分）**   通过Buffer、Len和Key参数分别接收主函数提供的密文字符串、密文实际长度和密钥，输出解密后的明文字符串。   1. **主函数int main(void)（5分）**   依次执行以下操作：输入明文字串和密钥；加密明文并输出；根据密文猜测密钥并输出；解密并输出明文。   1. **编程限制**   编程时，此题禁止使用string.h中的任何字符串操作函数。  ***[输入示例]***  *High-sensitivity X-ray polarimetry is a window not yet opened in astronomy due to technical difficulties. PolarLight, with a miniature X-ray polarimeter onboard a CubeSat, is a space program to test a novel technique for X-ray polarimetry. The CubeSat was launched into orbit on October 29, 2018. On December 18, the detector was powered on for in-orbit test, and detected the first events triggered by cosmic X-rays and charged particles.*  *4*  ***[输出示例]***  *Lmkl-wirwmxmzmxc B-vec tspevmqixvc mw e amrhsa rsx cix stirih mr ewxvsrsqc hyi xs xiglrmgep hmjjmgypxmiw. TspevPmklx, amxl e qmrmexyvi B-vec tspevmqixiv srfsevh e GyfiWex, mw e wtegi tvskveq xs xiwx e rszip xiglrmuyi jsv B-vec tspevmqixvc. Xli GyfiWex aew peyrglih mrxs svfmx sr Sgxsfiv 29, 2018. Sr Higiqfiv 18, xli hixigxsv aew tsaivih sr jsv mr-svfmx xiwx, erh hixigxih xli jmvwx izirxw xvmkkivih fc gswqmg B-vecw erh glevkih tevxmgpiw.*  *4*  *High-sensitivity X-ray polarimetry is a window not yet opened in astronomy due to technical difficulties. PolarLight, with a miniature X-ray polarimeter onboard a CubeSat, is a space program to test a novel technique for X-ray polarimetry. The CubeSat was launched into orbit on October 29, 2018. On December 18, the detector was powered on for in-orbit test, and detected the first events triggered by cosmic X-rays and charged particles.*  *请在此处粘贴运行结果截图*     1. **数组处理（30分 + 15分附加分）**   矩阵的梯度和余弦相似性的计算具有重要的实际应用价值。  给定一个的矩阵，其横向梯度表示了该矩阵的元素在横向坐标上的变化，一般计算方法为：在某一行内，第0列的横向梯度值就是第1列元素值减去第0列的元素值除以1；第1列的横向梯度值就是第1列元素值减去第0列的元素值，然后加上第2列元素值减去第1列元素值，最后再除以2；第i列（1≤i≤-2）的横向梯度值就是第i列元素值减去第i-1列的元素值，然后加上第i+1列元素值减去第i列元素值，最后再除以2；最后一列的横向梯度值是最后一列元素值减去倒数第二列元素值后除以1。矩阵的纵向梯度值有类似的计算。例如：矩阵为  则其横向和纵向梯度矩阵分别为：  给定一个的矩阵，矩阵的第i行为一个的向量。对于任意两个行向量和，它们之间的余弦相似性被定义为：  若其中某个行向量的所有元素全为0，则相似性也为0。相似性的值是介于-1到1之间的小数。对于上述矩阵，它的行向量对应的余弦相似性矩阵为：  其中第i行k列的元素表示行向量和的余弦相似性。  请按照要求编写一个程序，实现矩阵的运算操作。  **编程要求如下（函数参数与返回值可自拟）**：   1. **函数Input()（5分）**   该函数完成原矩阵行列数和各个元素值（可能非整数）的输入，并传回给主函数。   1. **函数MatrixGradientX()** **（10分）**   该函数接收原矩阵，完成矩阵横向梯度值的计算，生成横向梯度矩阵并返回给主函数。   1. **函数MaxGradientX()** **（5分）**   找出矩阵所有横向梯度绝对值最大的元素，并将这些元素的下标值和元素值清晰输出。   1. **函数MatrixSim() （附加分：10分）**   该函数计算的余弦相似性矩阵，并将传回主函数。   1. **函数ListSim() （附加分：5分）**   该函数首先输入一个合法行下标i，计算除之外的其他行向量与之间的余弦相似性，并按余弦相似性从高到低排序，从屏幕输出相应的行下标和相似性值（保留三位小数）。   1. **函数print() （5分）**   依次输出矩阵的所有元素，统一保留三位小数。   1. **主函数int main() （5分）**   调用input输入矩阵的行列数和各个元素，调用print输出矩阵；  调用MatrixGradientX函数计算横向梯度矩阵，调用print输出；  调用MaxGradientX函数输出横向梯度矩阵中绝对值最大所有元素下标和原矩阵中对应元素的值（保留三位小数）；  （附加分）调用MatrixSim函数计算余弦相似性矩阵，调用print输出；  （附加分）调用ListSim函数根据输入的行下标，输出按余弦相似性排序后的结果。  上述五个步骤的输出之间需有且只有一个空行。  ***[输入示例]***  *4 5*  *3 5 4 3 0*  *4 4 4 2 3*  *2 4 3 3 5*  *4 5 0 3 4*  *1*  ***[输出示例]***  *3.000 5.000 4.000 3.000 0.000*  *4.000 4.000 4.000 2.000 3.000*  *2.000 4.000 3.000 3.000 5.000*  *4.000 5.000 0.000 3.000 4.000*  *2.000 0.500 -1.000 -2.000 -3.000*  *0.000 0.000 -1.000 -0.500 1.000*  *2.000 0.500 -0.500 1.000 2.000*  *1.000 -2.000 -1.000 2.000 1.000*  *0 4 0.000*  *1.000 0.900 0.771 0.737*  *0.900 1.000 0.919 0.851*  *0.771 0.919 1.000 0.884*  *0.737 0.851 0.884 1.000*  *2 0.919*  *0 0.900*  *3 0.851*  *请在此处粘贴运行结果截图*     1. **格雷码转换 （30分）**   格雷码(Gray code)又叫循环二进制码或反射二进制码，是一种错误最小化的二进制编码方式，如下是十进制、4位自然二进制码与4位格雷码的对照表。   |  |  |  | | --- | --- | --- | | 十进制数 | 自然二进制数 | 格雷码 | | 0 | 0000 | 0000 | | 1 | 0001 | 0001 | | 2 | 0010 | 0011 | | 3 | 0011 | 0010 | | 4 | 0100 | 0110 | | 5 | 0101 | 0111 | | 6 | 0110 | 0101 | | 7 | 0111 | 0100 | | 8 | 1000 | 1100 | | 9 | 1001 | 1101 | | 10 | 1010 | 1111 | | 11 | 1011 | 1110 | | 12 | 1100 | 1010 | | 13 | 1101 | 1011 | | 14 | 1110 | 1001 | | 15 | 1111 | 1000 |   事实上，二进制数和格雷码之间存在如下的快速转换方法（下图中，b3代表B的最高位，以此类推）：   * 二进制数→格雷码转换快速方法（G：格雷码 B：二进位码）   G(i) = B(i+1) ⊕ B(i)     * 格雷码→二进制数   B(i) = B(i+1) ⊕ G(i)    请按照要求编写程序，实现格雷码和二进制数的互相转换。  **编程要求如下：**   1. **函数Input()** **（函数参数与返回值可自拟）** **（10分）**   该函数从键盘输入一个十进制数（范围0-15），并将其转化成四位二进制数，以字符串s的形式传回主函数，s的元素由ASCII码的0、1数字组成，高位在前（字符串头部）、低位在后（字符串尾部），如”0011”。   1. **函数void gray(char \*s, int flag)**  **（10分）**   若flag=1，输入的s为二进制码，gray将s转换成格雷码，传回主函数；（5分）  若flag=-1，输入的s为格雷码，gray将s转换成二进制码，传回主函数。（5分）   1. **主函数int main()** **（10分）**   程序循环输入并执行以下操作：  首先输入一个字母，该字母为g时，执行格雷码转二进制数操作；该字母为b时，执行二进制数转格雷码操作；否则，退出程序。  调用函数input()输入指定范围内的十进制数（不考虑范围之外的情况），并返回四位字符串；  调用函数gray执行转换操作；  输出转换后的值。  ***[输入示例]***  *b*  *10*  *g*  *15*  *q*  ***[输出示例]***  *1111*  *1010*  *请在此处粘贴运行结果截图*     1. **递归编程（10分）**   给定一个字符串，其中可能包含字母、数字、空格及其它等各种字符，要求判断该字符串中’(‘和’)’小括号是否匹配，即每个’)’在它前面是否都有唯一一个’(’与之对应。特别地，我们认为空字符串””和任意不包含’(’和’)’的字符串均为合法字符串，认为其是匹配的。例如：  1. "()" 匹配；  2. ")(" 不匹配；  3. "(abcd(e)" 不匹配；  4. "(a)(b)" 匹配。  5. "abc" 匹配  现要求必须使用**递归函数**实现上述字符串中括号是否匹配的判断，程序输入**两行**字符串str1和str2（每行字符串长度不超过80），首先对两行字符串进行**拼接**，将第二行的字符串str2拼接到第一行的字符串str1后面组成一个新的字符串str\_new，然后判断新的字符串str\_new中小括号是否匹配，如果匹配则输出”**Yes**”，否则输出”**No**”。  请按照要求编写程序，实现合并后的字符串中小括号是否匹配的判断。  **编程要求如下：**   1. **函数 Input() （函数参数与返回值可自拟）（3分）**   从键盘输入待拼接的两行字符串str1和str2（长度均不超过80），将其拼接之后组成新的字符串str\_new返回给主函数。   1. **递归函数 Is\_match ()（函数参数与返回值可自拟）（5分）**   使用递归的方法，对字符串中小括号是否匹配进行判断，并将判断的结果传回给主函数。   1. **主函数 int main()（2分）**   主函数调用Input()函数输入待拼接的两行字符串str1和str2，得到拼接后的字符串str\_new；然后调用递归函数Is\_match ()判断新字符串str\_new中小括号是否匹配；最后在主函数中输出匹配的结果。  ***[输入示例1]***  *abc(d)(*  *)ghj(d)*  ***[输出示例1]***  *Yes*  ***[输入示例2]***  *((()*  *)))*  ***[输出示例2]***  *No*  *请在此处粘贴运行结果截图*    *再如*    //以上为全部题目。 |