## 循环首次适应算法

任务：实现操作系统中存储分配的空闲分区链算法：循环首次适应算法。该算法是由首次循环算法演变而来，空闲区链仍按起始地址递增顺序排列只是不再每次从链首开始查找，而是从上次找到的空闲区的下一个空闲区开始查找，直到找到第一个能满足要求的空闲分区，并从中划出一块大小相等的内存空间分配给作业。若最后一个（链尾）空闲区，其大小仍不能满足要求，则返回链首继续查找。本算法的特点是空闲区分布均匀，大空闲区少。与首次适应算法相比，该算法减少了查找的开销，但也造就了大空闲区的缺乏。

要求： 模拟实现分配过程。

## 最佳适应算法

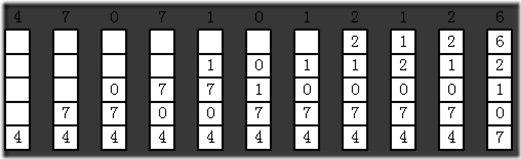
任务：实现操作系统中存储分配的空闲分区链算法：最佳适应算法。空闲区按分区大小递增顺序链接，该算法每次找到的空闲区都是最小满足要求的分区。然而，每次分配后所留下的部分总是最小的，这样，在存储器中会留下许多难以利用的小空闲区。特点，碎片多而小。要求： 模拟实现分配过程。

## 最近最久未使用LRU置换算法

任务：在虚拟存储器的页面置换算法中，比较流行的最近最久未使用LRU置换算法，就是利用栈来实现的。它利用一个特殊的栈来保存当前使用的各个页面的页面号。每当进程访问某页面时，便将该页面的页面号从栈中移出，将它压入栈顶。因此，栈顶始终是最新被访问页面的页面号，而栈底则是最近最久未被使用页面的页面号。

假定现有一进程所访问页面的页面号序列为：4，7，0，7，1，0，1，2，1，2，6随着进程的访问，栈中页面号的情况如下图所示。在访问页面时发生6次缺页，此时页面4是最近最久未被访问的页，应将它置换出去。

要求： 模拟实现调度过程。



## 考试报名管理

任务：考试报名工作给各高校报名工作带来了新的挑战， 给教务管理部门增加了很大的工作量，报名数据手工录入既费时又会不可避免地出现错误。本项目是对考试报名管理的简单模拟，用菜单选择方式完成下列功能：输入考生信息；输出考生信息；查询考生信息；添加考生信息；修改考生信息；删除考生信息。

要求：分别使用顺序表、链表、二叉排序树、索引表完成上述功能。

## 约瑟夫双向生死游戏

任务：约瑟夫双向生死游戏是在约瑟夫生者死者游戏的基础上，正向计数后反向计数，然后再正向计数。具体描述如下：30个旅客同乘一条船，因为严重超载，加上风高浪大，危险万分；因此船长告诉乘客，只有将全船一半的旅客投入海中，其余人才能幸免遇难。无奈，大家只得同意这种办法，并议定 30 个人围成一圈，由第一个人开始，顺时针依次报数，数到第 9人，便把他投入大海中，然后从他的下一个人数起，逆时针数到第 5 人，将他投入大海，然后从他逆时针的下一个人数起，顺时针数到第 9 人，再将他投入大海，如此循环，直到剩下15 个乘客为止。

要求：求出哪些位置是将被扔下大海的位置。动态演示求解过程。

## 软件工程进度规划

任务：设计一个软件， 需要进行用户需求分析、 系统需求确认、 系统概要设计、 设计用例场景、系统的详细设计、数据库详细设计、编码、单元测试、集成测试、系统测试、维护等活动。用户需求分析需要在系统需求确认之前完成， 系统的系统的详细设计必须在系统的概要设计、设计系统用例和设计用例场景之前完成。如表所示是一系列活动之间的关系。

|  |  |  |
| --- | --- | --- |
| 活动代码 | 活动名称 | 先需活动 |
| A1 | 用户需求分析 | 无 |
| A2 | 系统需求确认 | A1 |
| A3 | 系统概要设计 | A2 |
| A4 | 设计用例场景 | 无 |
| A5 | 系统的详细设计 | A3, A4 |
| A6 | 数据库详细设计 | A3 |
| A7 | 编码 | A5, A6 |
| A8 | 单元测试 | A7 |
| A9 | 集成测试 | A8 |
| A10 | 系统测试 | A7 |
| A11 | 维护 | A11 |

要求：请设计算法判断该软件设计流程是否有回路，若无，请给出该软件设计 AOV 网的拓朴序列。

## 贪吃蛇

任务：贪吃蛇的核心数据结构其实就是“Queue”，蛇就是一个queue，蛇吃食物其实就是往“蛇头“push一个格子；而蛇的移动，其实就是把”蛇尾“pop，然后”根据移动方向“向”蛇头“push。游戏的区域，是一个二维的矩形，其中的每一个格子，可以用(x, y)表示，而矩形的长宽，有最大值；区域中的每个格子，要么是”墙“，要么是”蛇“的一部分，要么就是”食物“（还有一种，就是“空白”）

要求：利用队列实现上述功能。

## 八皇后问题

任务：求出在一个n×n的棋盘上，放置n个不能互相捕捉的国际象棋“皇后”的所有布局。

这是来源于国际象棋的一个问题。皇后可以沿着纵横和两条斜线8个方向相互捕捉。如图所示，一个皇后放在棋盘的第4行第3列位置上，则棋盘上凡打“×”的位置上的皇后就能与这个皇后相互捕捉，也就是下一个皇后不能放的位置。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  | × |  |  | × |  |  |
| × |  | × |  | × |  |  |  |
|  | × | × | × |  |  |  |  |
| × | × | Q | × | × | × | × | × |
|  | × | × | × |  |  |  |  |
| × |  | × |  | × |  |  |  |
|  |  | × |  |  | × |  |  |
|  |  | × |  |  |  | × |  |

从图中可以得到以下启示：一个合适的解应是在每列、每行上只有一个皇后，且一条斜线上也只有一个皇后。

要求：求解过程从空配置开始。在第1列至第m列为合理配置的基础上，再配置第m+1列，直至第n列配置也是合理时，就找到了一个解。接着改变第n列配置，希望获得下一个解。另外，在任一列上，可能有n种配置。开始时配置在第1行，以后改变时，顺次选择第2行、第3行、…、直到第n行。当第n行配置也找不到一个合理的配置时，就要回溯，去改变前一列的配置。

## 强连通分量

任务：对于输入的有向图，实现深度优先遍历、广度优先遍历。判断该有向图是否强连通，若不是强连通图则求出该图的所有强连通分量并输出。

要求：按要求完成上述功能。

## 栈的应用—逆波兰式求值

任务：掌握栈的特点及其描述方法；掌握栈的各种基本操作；掌握栈的一个经典应用;逆波兰式求值问题。

要求：从键盘输入一个整数表达式，先将其转化为逆波兰表达式，再计算值。

## 哈夫曼编码/译码器

任务：设计一个利用哈夫曼算法的编码和译码系统，重复地显示并处理以下项目，直到选择退出为止。

要求：将权值数据存放在数据文件(文件名为data.txt，位于执行程序的当前目录中) 分别采用动态和静态存储结构。

初始化：键盘输入字符集大小n、n个字符和n个权值，建立哈夫曼树；

编码：利用建好的哈夫曼树生成哈夫曼编码；输出编码；

译码功能；显示哈夫曼树；

设字符集及频度如下表：

字符 空格 A B C D E F G H I J K L M

频度 186 64 13 22 32 103 21 15 47 57 1 5 32 20

字符 N O P Q R S T U V W X Y Z

频度 57 63 15 1 48 51 80 23 8 18 1 16 1

界面设计的优化。

## 简易的社交网络图

任务：图的应用—简易的社交网络图

要求：掌握图的几种存储方法（邻接矩阵、邻接表等）掌握图的连通和图中各节点的联系。 输入A,B,C,D,E,F处的人名和权值，计算某两人之间的陌生度（即权值的大小之和）与连通两人的通路。直接相连的两个节点间边的权值就是两节点间的亲密度（陌生度），权值越小，节点间越亲密；通过邻接矩阵给出的数值可以求出两节点间的亲密度；通过迪克斯特卡算法可以求出不相邻的两节点间的权值之和和经过的节点，从而达到解决问题的目的。

## 优先队列

任务：优先队列是一种用来维护由一组元素构成的集合S的数据结构，其中的每一个元素都有一个相关的值，称为关键字。优先队列也分为两种：最大优先队列和最小优先队列。

一个最大优先队列支持以下操作：

•INSERT(S,x)：把元素x插入集合S中;

•MAXIMUM(S)：返回S中具有最大关键字的元素;

•EXTRACT\_MAX(S)：去掉并且返回S中的具有最大关键字的元素;

•INCREASE\_KEY(S,x,k)：将元素x的关键字值增加到k。

相应地，最小优先队列支持的操作包括INSERT、MINIMUM、EXTRAT\_MIN和DECRESE\_KEY。

要求：实现优先队列，及各种操作。

## 堆排序的改进算法

任务：堆是计算机科学中一类特殊的数据结构的统称。堆通常是一个可以被看做一棵树的数组对象。

要求：查阅相关文献，寻找改进的堆排序算法，实现并与原排序算法进行比较。

## 客户消费积分管理系统

任务：针对客户的消费情况，进行客户管理，根据客户的消费积分对客户实行不同程度的打折优惠。

要求：

采用一定的存储结构进行客户信息的存储；

对客户的信息可以进行修改、删除、添加；

能够根据消费情况进行客户积分的计算；

根据积分情况实行不同程度的打折优惠；

## 排序综合

任务：利用随机函数产生N个随机整数（20000以上），对这些数进行多种方法进行排序。

要求：

1)至少采用三种方法实现上述问题求解（提示，可采用的方法有插入排序、希尔排序、起泡排序、快速排序、选择排序、堆排序、归并排序）。并把排序后的结果保存在不同的文件中。

2)统计每一种排序方法的性能（以上机运行程序所花费的时间为准进行对比），找出其中两种较快的方法。

## 学生搭配问题

任务：一班有m个女生,有n个男生(m不等于n),现要开一个舞会. 男女生分别编号坐在舞池的两边的椅子上.每曲开始时,依次从男生和女生中各出一人配对跳舞, 本曲没成功配对者坐着等待下一曲找舞伴.

要求：请设计一系统模拟动态地显示出上述过程,要求如下:

1)输出每曲配对情况

2)计算出任何一个男生(编号为X)和任意女生(编号为Y),在第K曲配对跳舞的情况.至少求出K的两个值.

3)尽量设计出多种算法及程序,可视情况适当加分

 提示:用队列来解决比较方便.

## 猴子吃桃子问题

任务：有一群猴子摘了一堆桃子，他们每天都吃当前桃子的一半且再多吃一个，到了第10天就只余下一个桃子。用多种方法实现求出原来这群猴子共摘了多少个桃子。

要求：

1)采用数组数据结构实现上述求解

2)采用链数据结构实现上述求解

3)采用递归实现上述求解

## 稀疏矩阵运算器

任务：稀疏矩阵是指那些多数元素为零的矩阵。利用 " 稀疏 " 特点进行存储和计算可以大大 节省存储空间 , 提高计算效率。实现一个能进行稀疏矩阵基本运算的运算器。以"带行逻辑链接信息"的三元组顺序表表示稀疏矩阵,实现两个矩阵相加、相减和相乘的运算。稀疏矩阵的输入形式采用三元组表示 , 而运算结果的矩阵则以通常的阵列形式列出。

要求：1. 首先应输入矩阵的行数和列数 , 并判别给出的两个矩阵的行、列数对于所要求作

的运算是否相匹配。可设矩阵的行数和列数均不超过 20 。2. 程序可以对三元组的输入顺序加以限制 , 例如 , 按行优先。3. 在用三元组表示稀疏矩阵时, 相加或相减所得结果矩阵应该另生成 , 乘积矩阵也可用二维数组存放。4. 按教科书 5.3.2 节中的描述方法 , 以十字链表表示稀疏矩阵。5. 矩阵求逆的运算 , 包括不可求逆的情况。在求逆之前 , 先将稀疏矩阵的内部表示改为十字链表。

## 哈希表设计

任务：针对某个集体(比如你所在的班级)中的"人名"设计一个哈希表，使得平均查找长度不超过R，完成相应的建表和查表程序。假设人名为中国人姓名的汉语拼音形式。待填入哈希表的人名共有30个，取平均查找长度的上限为2。哈希函数用除留余数法构造，用伪随机探测再散列法处理冲突。测试数据取读者周围较熟悉的30个人的姓名。

要求：如果随机函数自行构造，则应首先调整好随机函数，使其分布均匀。人名的长度均不超过19个字符(最长的人名如z庄双双(ZhangStmarlgShang〉。字符的取码方法可直接利用C语言中的toascii函数，并可对过长的人名先作折叠处理。从教科书上介绍的几种哈希函数构造方法中选出适用者并设计几个不同的哈希函数，比较它们的地址冲突率(可以用更大的名字集合作试验。研究人名的特点，努力找一个哈希函数，使得对于不同的拼音名一定不发生地址冲突。在哈希函数确定的前提下尝试各种不同处理冲突的方法，考查平均查找长度的变化和造好的哈希表中关键字的聚簇性。

## 图书管理

任务：图书管理基本业务活动包括:对一本书的采编入库、清除库存、借阅和归还等等。试设计一个图书管理系统，将上述业务活动借助于计算机系统完成。

要求：〈1〉每种书的登记内容至少包括书号、书名、著者、现存量和总库存量等五项。

〈2〉作为演示系统，不必使用文件，全部数据可以都在内存存放。但是由于上述四项基本业务活动都是通过书号(即关键字〉进行的，所以要用B树〈24树〉对书号建立索引，以获得高效率。

〈3〉系统应实现的操作及其功能定义如下:

采编入库z新购入一种书，经分类和确定书号之后登记到图书账目中去。如果这种书在账中已有，则只将总库存量增加。

② 清除库存:某种书已无保留价值，将它从图书账目中注销。

③ 借阅:如果一种书的现存量大于零，则借出一本，登记借阅者的图书证号和归还期限。

④ 归还z注销对借阅者的登记，改变该书的现存量。

⑤ 显示:以凹入表的形式显示B树。这个操作是为了调试和维护的目的而设置的。

【测试数据】

入库书号:35，16，18，70，5，50，22，60，13，17，12，45，25，毡，15，90，30，7然后清除:45，90，50，22，42

其余数据自行设计。由空树开始，每插入删除一个关键字后就显示B树的状态。

【实现提示】

(1)24树的查找算法是基础，入库和清除操作都要调用。难点在于删除关键字的算法，因而只要算法对2-3树适用就可以了，暂时不必追求高阶B树也适用的删除算法。

(2)每种书的记录可以用动(或静)态链式结构。借阅登记信息可以链接在相应的那种书的记录之后。

(l)将一次会话过程(即程序一次运行)中的全部人机对话记入一个日志文件"log"中去。

(2)增加列出某著者全部著作名的操作。思考如何提高这一操作的效率，参阅教科书12.6.2节。

(3)增加列出某种书状态的操作。状态信息除了包括这种书记录的全部信息外还包括最早到期(包括已逾期)的借阅者证号，日期可用整数实现，以求简化。

(4)增加预约借书功能。

## 平衡二叉树操作的演示

任务：利用平衡二叉树实现一个动态查找表。实现动态查找表的三种基本功能:查找、插入和删除。

要求：(1)初始，平衡二叉树为空树，操作界面给出查找、插入和删除三种操作供选择。每种操作均要提示输入关键字。每次插入或删除一个结点后，应更新平衡二叉树的显示。 (2)平衡二叉树的显示可采用凹入表形式，也可以采用图形界面画出树形。(3)教科书已给出查找和插入算法，本题重点在于对删除算法的设计和实现。假设要删除关键字为x的结点。如果x不在叶子结点上，则用它左子树中的最大值或右子树中的最小值取代x。如此反复取代，直到删除动作传递到某个叶子结点。删除叶子结点时，若需要进行平衡变换，可采用插入的平衡变换的反变换(如，左子树变矮对应于右子树长高)。(4)合并两棵平衡二叉树。(5)把一棵平衡二叉树分裂为两棵平衡二叉树，使得在一棵树中的所有关键字都小于或等于工，另一棵树中的任一关键字都大于几。

## 英语词典的维护和识别

任务：Trie树通常作为一种索引树，这种结构对于大小变化很大的关键字特别有用。利用Trie树实现一个英语单词辅助记忆系统，完成相应的建表和查表程序。

要求：不限定Trie树的层次，每个叶子结点只含一个关键字，采用单字符逐层分割的策略，实现Trie树的插入、删除和查询的算法，查询可以有两种方式:查询一个完整的单词或者查询以某几个字母开头的单词。以实习三中已实现的串类型或C语言中提供的长度不限的串类型表示关键字，叶子结点内应包括英语单词及其注音、释义等信息。限定Trie树的层次，每个叶子结点可以包含多个关键字。

## 内部排序算法比较

任务：在教科书中，各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时间。试通过随机数据比较各算法的关键字比较次数和关键字移动次数，以取得直观感受。

要求：(1)对以下6种常用的内部排序算法进行比较z起泡排序、直接插入排序、简单选择排序、快速排序、希尔排序、堆排序。(2)待排序表的表长不小于1005其中的数据要用伪随机数产生程序产生：至少要用5组不同的输入数据作比较：比较的指标为有关键字参加的比较次数和关键字的移动次数(关键字交换计为3次移动)。(3)最后要对结果作出简单分析，包括对各组数据得出结果波动大小的解释。测试数据由随机数产生器生成。

主要工作是设法在已知算法中的适当位置插入对关键字的比较次数和移动次数的计数操作。程序还可以考虑几组数据的典型性，如，正序、逆序和不同程度的乱序。注意采用分块调试的方法。实现折半插入排序、二路插入排序、归并排序、基数排序。对不同的输入表长作试验，观察检查两个指标相对于表长的变化关系。还可以对稳定性作验证。

## 多关键字排序

任务：多关键字的排序有其一定的实用范围。例如:在进行高考分数处理时，除了需对总分进行排序外，不同的专业对单科分数的要求不同，因此尚需在总分相同的情况下，按用户提出的单科分数的次序要求排出考生录取的次序。

要求：

(1)假设待排序的记录数不超过10000，表中记录的关键字数不超过5，各个关键字的范围均为0至100。按用户给定的进行排序的关键字的优先关系，输出排序结果。

(2)约定按LSD法进行多关键字的排序。在对各个关键字进行排序时采用两种策略:其一是利用稳定的内部排序法，其二是利用"分配"和"收集"的方法。并综合比较这两种策略。

测试数据由随机数产生器生成。

（3）按MSD策略进行排序，并和上述两种排序策略进行综合比较。

实现提示：用5至8组数据比较不同排序策略所需时间。由于是按LSD方法进行排序，则对每个关键字均可进行整个序列的排序，但在利用通常的内部排序方法进行排序时，必须选用稳定的排序方法。借助"分配"和"收集"策略进行的排序，如同一趟"基数排序"，由于关键字的取值范围为0至100，则分配时将得到104个链表。

## 文件索引

任务：已知职工文件中包括职工号、职工姓名、职务和职称等若干数据项(见下表)。职务有校长、系主任、室主任和教员；校长领导所有系主任，系主任领导他所在系的所有室主任，室主任领导他所在室的全体教员；职称有教授、副教授和讲师3种。

请给该文件建立索引,通过该索引文件,能做到:

(1)能够检索出全体职工间领导与被领导的情况；

(2)能够分别检索出全体教授、全体副教授、全体讲师。要求指针数量尽可能少，给出各指针项索引的名称及含义即可

|  |  |  |  |
| --- | --- | --- | --- |
| 职工号 | 职工姓名 | 职务 | 职称 |
| 1 | 张军 | 教员 | 讲师 |
| 2 | 沈灵 | 系主任 | 教授 |
| 3 | 叶明 | 校长 | 教授 |
| 4 | 张莲 | 室主任 | 副教授 |
| 5 | 叶宏 | 系主任 | 教授 |
| 6 | 周芳 | 教员 | 教授 |
| 7 | 刘光 | 系主任 | 教授 |
| 8 | 黄兵 | 教员 | 讲师 |
| 9 | 李民 | 室主任 | 教授 |
| 10 | 赵松 | 教员 | 副教授 |
| … | … | … | … |

## 成绩分析问题

任务：录入、保存一个班级学生多门课程的成绩，并对成绩进行分析。

要求：1、通过键盘输入各学生的多门课程的成绩，建立相应的文件input.dat。

2、对文件input.dat中的数据进行处理，要求具有如下功能：

(1)按各门课程成绩排序，并生成相应的文件输出。

(2)计算每人的平均成绩，按平均成绩排序，并生成文件。

(3)求出各门课程的平均成绩、最高分、最低分、不及格人数、60~69分人数、70~79分人数、80~89分人数、90分以上人数。

(4)根据姓名或学号查询某人的各门课成绩，重名情况也能处理。

3、界面美观。

## 城市链表

任务：将若干城市的信息，存入一个带头结点的单链表。结点中的城市信息包括：城市名，城市的位置坐标。要求能够利用城市名和位置坐标进行有关查找、插入、删除、更新等操作。

要求：（1） 给定一个城市名，返回其位置坐标；（2） 给定一个位置坐标P和一个距离D，返回所有与P的距离小于等于D的城市。

## 运动会分数统计

任务：参加运动会的n个学校编号为1～n。比赛分成m个男子项目和w个女子项目，项目编号分别为1～m和m+1～m+w。由于各项目参加人数差别较大，有些项目取前五名，得分顺序为7，5，3，2，1；还有些项目只取前三名，得分顺序为5，3，2。写一个统计程序产生各种成绩单和得分报表。

要求：可以输入各个项目的前三名或前五名的成绩；能统计各学校总分，可以按学校编号或名称、学校总分、男女团体总分排序输出；可以按学校编号查询学校某个项目的情况；可以按项目编号查询取得前三或前五名的学校。数据存入文件并能随时查询。

规定：输入数据形式和范围：可以输入学校的名称，运动项目的名称

输出形式：有中文提示，各学校分数为整型。

界面要求：有合理的提示，每个功能可以设立菜单，根据提示，可以完成相关的功能要求。

存储结构：学生自己根据系统功能要求自己设计，但是要求运动会的相关数据要存储在数据文件中。

测试数据：

要求使用1、全部合法数据；2、整体非法数据；3、局部非法数据。进行程序测试，以保证程序的稳定。

例如，对于n=4，m=3，w =2，编号为奇数的项目取前五名，编号为偶数的项目取前三名，设计一组实例数据。

实现提示：

可以假设n≤20，m≤30，w≤20，姓名长度不超过 20 个字符。每个项目结束时，将其 编号、类型符(区分取前五名还是前三名) 输入，并按名次顺序输入运动员姓名、校名(和成绩)。

## 约瑟夫环

任务：约瑟夫 (Joseph) 问题的一种描述是：编号为 1，2，… ，n 的n个人按顺时针方向围坐一 圈，每人持有一个密码(正整数)。一开始任选一个正整数作为报数上限值m，从第一个人开始按顺时针方向自1开始顺序报数，报到m时停止报数。报m的人出列，将他的密码作为新的m值，从他在顺时针方向上的下一个人开始重新从1报数，如此下去，直至所有人全部出列为止。试设计一个程序求出出列顺序。

要求：利用单向循环链表存储结构模拟此过程，按照出列的顺序印出各人的编号。向上述程序中添加在顺序结构上实现的部分。

测试数据：m的初值为 20；n=7，7 个人的密码依次为：3，1，7，2，4，8，4， 首先 m 值为 6( 正确的出列顺序应为 6，1，4，7，2，3，5) 。

实现提示：程序运行后，首先要求用户指定初始报数上限值，然后读取各人的密码。可设n≤30。 此题所用的循环链表中不需要 “头结点”，请注意空表和非空表的界限。

## 重言式判别

任务：一个逻辑表达式如果对于其变元的任一种取值都为真，则称为重言式；反之，如果对于其变元的任一种取值都为假，则称为矛盾式；然而，更多的情况下，既非重言式，也非矛盾式。试写一个程序，通过真值表判别一个逻辑表达式属于上述哪一类。

要求：(1) 逻辑表达式从终端输入，长度不超过一行。逻辑运算符包括 "|"，"&" 和 "~"，分别表示或、与和非，运算优先程度递增，但可由括号改变，即括号内的运算优先。逻辑变元 为大写字母。表达式中任何地方都可以含有多个空格符。

(2) 若是重言式或矛盾式，可以只显示"True forever"，或"False forever"，否则显示 "Satisfactible" 以及变量名序列，与用户交互。若用户对表达式中变元取定一组值，程序就求出并显示逻辑表达式的值。

测试数据：

(1) (A|~A)&(B|~B)

(2) (A&~A)&C

(3) A|B|C|D|E|~A

(4) A&B&C&~B

(5) (A|B)&(A|~B)

(6) A&~B|~A&B;O ,0;0,1;1,0;1,1 。

## 长整数四则运算

任务：设计一个实现任意长的整数进行加法运算的演示程序。

要求：利用双向循环链表实现长整数的存储，每个结点含一个整型变量。任何整型变量的范 围是－(215-l)~(215-1) 。输入和输出形式：按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开。实现长整数的四则运算；(2) 实现长整数的乘方和阶乘运算；(3) 整型量范围是－ (2n－1) ~ (2n－1)， 其中，n是由程序读入的参量。输入数据的分组方法可以另行规定。

测试数据：

(1) 0；0； 应输出 "0" 。

(2)－2345，6789； －7654，3211； 应输出 "－1，0000，0000" 。

(3)－9999，9999；1，0000，0000，0000； 应输出 "9999，0000，0001" 。

(4) 1，0001，0001；－1，0001，0001； 应输出 "0" 。

(5) 1，0001，0001；－1，0001，0000； 应输出 "1" 。

(6) -9999，9999，9999；－9999，9999，9999；应输出 "－1，9999，9999，9998" 。 (7) 1，0000，9999，9999；1； 应输出 "1，0001，0000，0000 " 。

实现提示：

(1) 每个结点中可以存放的最大整数为 215-1=32767， 才能保证两数相加不会溢出。但若这样存放，即相当于按32768进制数存放，在十进制数与32768进制数之间的转换十分不方便。故可以在每个结点中仅存十进制数的4位，即不超过9999的非负整数，整个链表表示为万进制数。

(2) 可以利用头结点数据域的符号代表长整数的符号。相加过程中不要破坏两个操作数链表。不能给长整数位数规定上限。

## 一元稀疏多项式计算器

任务：设计一个一元稀疏多项式简单计算器。

要求：一元稀疏多项式简单计算器的基本功能是：

(1) 输入并建立多项式 ；

(2) 输出多项式，输出形式为整数序列：n，cl，el，c2，e2，…，cn，en，其中n是多项式的项数，ci 和ei，分别是第 i 项的系数和指数，序列按指数降序排列；

(3) 多项式a和b相加，建立多项式a +b；

(4) 多项式a和b相减，建立多项式a -b。

(5) 计算多项式在x处的值。

(6) 求多项式 a 的导函数 。

(7) 多项式a和b相乘，建立乘积多项式ab 。

(8) 多项式的输出形式为类数学表达式。例如 ，多项式 -3x8+6x3－18 的输出形式为

，x15+(－8)x7－14的输出形式为。注意，数值为1的非零次项的输出形式中略去系数1，如项1x8的输出形式为x8，项 －1x3的输出形式为－x3。

测试数据：

(1)(2x+5x8－3.1x11) + (7－5x8+11x9)=(－3.lx11+11x9+2x+7)

(2)(6x-3－x+4.4x2－1.2x9) －(-6x-3+5.4x2－x2+7.8x15)

=(-7.8x15-1.2x9+12x-3-x)

(3)(1 +x + x2+x3+x4+x5)+(-x3－x4)=(1+x+x2+x5)

(4)(x+x3)+(-x－x3)=0

(5)(x+x100)+(x100 +x200)=(x+2x100+x200)

(6)(x+x2+x3)+0=x+x2+x3

(7) 互换上述测试数据中的前后两个多项式

实现提示：用带表头结点的单链表存储多项式。

## 一致性哈希算法

题目：工程师经常使用服务器集群来设计和实现数据缓存，以下是常见的策略：

1)，无论是添加、查询还是删除数据，都是先将数据的id通过哈希函数转换成一个哈希值，记为key。

2),如果目前机器有n台，则计算key%n的值，这个值就是该数据所属的机器编号，无论是添加、删除还是查询操作，都只在这台机器上进行。

要求：请分析这种策略可能带来的问题，并提出改进的方案。

思路：[https://blog.csdn.net/u010075989/article/details/81117566，https://blog.csdn.net/suifeng629 /article/details/81567777](https://blog.csdn.net/u010075989/article/details/81117566，https://blog.csdn.net/suifeng629%20/article/details/81567777)

## 布隆过滤器

题目：我们经常会把一部分数据放在Redis等缓存，比如产品详情。这样有查询请求进来，我们可以根据产品Id直接去缓存中取数据，而不用读取数据库，这是提升性能最简单，最普遍，也是最有效的做法。一般的查询请求流程是这样的：先查缓存，有缓存的话直接返回，如果缓存中没有，再去数据库查询，然后再把数据库取出来的数据放入缓存。但是如果现在有大量请求进来，而且都在请求一个不存在的产品Id，那么大量的请求都发送到数据库，数据库的压力一下子就上来了。“布隆过滤器”就可以解决（缓解）缓存穿透问题。

要求：实现布隆过滤器

思路：<https://www.cnblogs.com/CodeBear/p/10911177.html，https://www.jb51.net/article/> 150190.htm，https://blog.csdn.net/yuanlaijike/article/details/94356729