## 全国交通咨询模拟

任务：处于对不同目的的旅客对交通工具有不同的要求。例如，因公出差的旅客希望在旅途中的时间尽可能短，出门旅游的游客则希望旅费尽可能省，而老年旅客则要求中转次数最少。编制一个全国城市间的交通咨询程序，为旅客提供两种或三种最优决策的交通咨询。

要求：（1）提供对城市信息进行编辑（如：添加或删除）的功能。（2）城市之间有两种交通工具：火车和飞机。提供对列车时刻表和飞机航班进行编辑（增设或删除）的功能。 （3）提供两种最优决策：最快到达或最省钱到达。全程只考虑一种交通工具。（4）旅途中耗费的总时间应该包括中转站的等候时间。（5）咨询以用户和计算机的对话方式进行。由用户输入起始站、终点站、最优决策原则和交通工具，输出信息：最快需要多长时间才能到达或者最少需要多少旅费才能到达，并详细说明依次于何时乘坐哪一趟列车或哪一次班机到何地。

测试数据：

徐州

397

乌鲁木齐

哈尔滨

长春

沈阳

天津

郑州

西安

兰州

成都

上海

昆明

贵阳

株州

福州

柳州

广州

深圳

南宁

1892

216

1145

668

1100

967

639

907

349

676

511

242

305

704

651

622

367

409

255

607

675

140

呼和浩特

北京

137

674

西宁

大连

534

842

武汉

825

南昌

672

实现提示：（1）对全国城市交通图和班车时刻表及飞机航班表的编辑，应该提供文件形式输入和键盘输入两种方式。飞机航班表的信息应包括：起始站的出发时间、终点站的到达时间和票价；列车时刻表则需根据交通图给出各个路段的详细信息，例如：对于从北京到上海的火车，需给出北京至天津、天津至徐州及徐州至各段的出发时间、到达时间和票价信息。（2）以邻接表作交通图的存储结构，表示边的结点内除含有邻接点的信息外，包括交通工具、路程中消耗的时间和花费以及出发和到达的时间等多项属性。

## 压缩器/解压器

任务：为了节省存储空间，常常需要把文本文件采用压缩编码的方式储存。例如：一个包含1000个x的字符串和2000个y的字符串的文本文件在不压缩时占用的空间为3002字节（每个x或每个y占用一个字节，两个字节用来表示串的结尾）。同样是这个文件，采用游程长度编码（run-length coding），可以存储为字符串1000x2000y，仅为10个字母，占用12个字节。若采用二进制表示游程长度（1000和2000）可以进一步节约空间。如果每个游程长度占用2个字节，则可以表示的最大游程长度为2\*pow(16)，这样，上例中的字符串只需要用8个字节来存储。当要读取编码文件时，需要对其进行解码。由压缩器（compressor）对文件进行编码，由解压器（decompressor）进行解码。

①（1）长度-游程编码的压缩/解压；+（2）LZW压缩/解压（散列）；

②（1）长度-游程编码的压缩/解压；+（3）霍夫曼编码压缩/解压 (霍夫曼树)

基本要求：要求选用二种压缩/解压策略实现压缩/解压器[（1）为必选]。输入的为本文文件（.txt），输出的为一种自定义的文件（.nz）。考虑当构成文本的字符集合为{a,b,c,……,z,0,1,2,…9}时，请用实例测试你的压缩/解压器。你的压缩器会不会出现抖动？（压缩后的文本比原来的还要大）。扩充构成文本的字符集合以便使它适应更一般的情况。

实现提示：LZW：由Lempel、Ziv和Welch这三位科学家所开发的技术。该方法把文本的字符串映射为编码，首先，为该文本中所有可能出现的字母分别分配一个代码。例如：要压缩的对象是aaabbbbbbbaabaaba，由a和b组成。为a分配代码0,为b分配代码1。字符串和编码的关系被存储在字典中。字典如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Code | A | b | Aa | aab | bb | bbb | bbba | aaba |

LZW压缩器不断的在输入文件中寻找在字典中出现的最长的前缀p,并输出其相应的代码。若输入文件的下一个字符为c，则为pc分配下一个代码，并插入字典，这种策略称为LZW规则。相反，在解压时，编码表由压缩文件重新构造，LZW原则使这种重建成为可能。

如上例子，压缩时，文件中第一个在字典中出现的最长前缀是a, 输出其编码0,然后为字符串aa分配代码2，并插入到字典中。余下的字符串在字典中出现的最长前缀是aa,输出aa的对应代码2，同时为字符串aab分配代码3并将其插入到字典中。依次类推，由此，输出0214537

解压时，要输入代码，然后用代码所表示的文本来替换这些代码。代码到文本的映射可按下面的方法重建：首先把分配给单一字母的代码插入到字典中。象前面一样，字典的入口为key-code对。然而此时是根据给定的代码（key）去寻找相应的入口（而不是根据文本Code）。压缩文件中的第一个代码对应于单一的字母，因此可以由该字母代替。对于压缩文件中的其他代码p,要考虑两种情况：1）在字典中；2）不在字典中。在1）情况下，找到p对应的文本text(p)输出。并且，根据压缩原理可知，若在压缩文件中代码q写在p之前且text(q)是与q对应的文本，则压缩器会为文本text(q)（其后紧跟fc(p)，text(p)的第一个字符）分配一新代码。因此在字典中插入序偶（下一个代码，text(q)fc(p)）。情况2）时，只有在当前文本段形如text(q)text(q)fc(q)且text(p)=text(q)fc(q)时才会发生。相应的压缩文件段是qp。在压缩过程中，为text(q)fc(q)分配的代码为p。在解压过程中，在用text(q)代替q后，又遇到代码p。然而，此时字典中没有与p对应的文本。因为这种情况只在解压文本段为text(p)text(q)fc(q)时才发生，因此可以对p解码。当遇到一个没有定义代码文本对的代码p时，p对应的文本为text(q)fc(q)，其中q为p前面的代码。

如上例子：首先，初始化字典，在其中插入(0,a),(1,b)。压缩的第一个代码为0,则用a代替之。下一个代码2未定义，因为前一个代码为0，且text(0)=a,fc(0)=a,则

text(2)=text(0)fc(0)=aa。因此用aa代替2,并把(2,aa)插入字典中。下个代码1由b来替换，并把(3,text(2)fc(1))=(3,aab)插入字典中。依次类推，得解压结果。

霍夫曼编码：根据不同符号在文本中出现的不同的频率来进行压缩编码。假设文本是由a,u,x,z组成的字符串，若这个字符串的长度为1000，每个字符用一个字节来存储，共需1000个字节（即8000位）的空间。如果每个字符用2位二进制来编码（00=a,01=x,10=u,11=z），则用2000位二进制即可以表示1000个字符。此外，还需要一定的空间来存放编码表，可以采用如下格式来存储：

符号个数：代码1，符号1，代码2，符号2，……

符号个数及每个符号分别用8位二进制来表示，每个代码需要占用[log2(符号个数)]位二进制。因此，上例中，代码表需占用5\*8+4\*2=48位，压缩比为8000/2048=3.9 。利用这种编码方法，字符串aaxuaxz的压缩码为二进制串00000110000111，每个字符的编码具有相同的位数（两位）。从左到右依次从位串中取出两位，通过查编码表边可以获得原字符串，这是解压缩过程。

我们利用霍夫曼编码来实现压缩，必须：

必须获得不同字符的频率。

建立具有最小加权外部路径的二叉数（即霍夫曼树），树的外部结点用字符串中

的字符表示，外部结点的权重（weight）即为该字符出现的频率。

遍历从根到外部结点的路径得到每个字符的编码。

使用字符的编码来代替字符串中的字符。

为了方便解码，需要保存字符代码映射表或每个字符的

频率表（在保存信息为频率表的情况下，解码需要重构霍夫曼数以获得相应的编码表）。

构造霍夫曼树：首先从仅含一个外部结点的二叉树集合开始，每个外部结点代表字符串的一个不同的字符，其权重等于该字符的频率。此后不断的从集合中选择两棵具有最小权重的二叉树，并把它们合并成一棵新的二叉树，合并方法是把这两棵二叉树分别作为左右子树，然后增加一个新的根结点。新二叉树的权重为两棵子树的权重之和。这个过程一直可以持续到仅剩下一棵树为止。[二叉树的集合可以使用有序表 /优先队列（基于队列/基于堆）实现]。

编码：构造完毕霍夫曼树后，可以对从根开始到外部结点（叶子）的路径进行编码，方法是向左孩子移动时取0，向右孩子移动时取1。

选作内容：对于策略2）我们用这种方法修改它：每当压缩/解压1024\*x个字节后，重新初始化代码表。取文本长度为100K到200K之间，x=10，20，30，40和50。测试修改后的程序，请给出你的结论：采用那种x值比较好？

对于霍夫曼编码：当文本中的字符出现的频率差别很大时，我们可以通过使用变长的编码来降低每个位串的长度。但是，怎样对使用变长编码的位串解码呢？我们可以发现：在得到的霍夫曼编码中，没有任何一个代码是另一个代码的前缀。因此与编码向匹配的实际的字符是唯一的。请用实现这样的变长策略，并验证它。

## 电视大赛观众投票及排名系统（排序应用）

任务：在很多的电视大赛中，通常当选手表演结束后，现场观众通过手中的按键对参赛选手进行投票，然后对选手获得的票数进行统计，从高到低进行降序排序，从而自动产生冠军、亚军和季军。现在要求编写一程序模拟实现上述系统的功能。

要求：在本例中，首先输入参赛选手的人数（范围为1-9个），然后根据人数通过malloc函数来开辟存放选手信息的顺序表。将选手的编号和姓名依此存入顺序表单元中，观众通过按键进行投票，按’1’为1号选手投票，按’2’为2号选手投票，以此类推，以按’0’作为投票结束标志。投票结束后进行排序，在此采用希尔排序，然后为每个选手计算名次，得票相同的名次也相同，存储类型的定义

参赛选手信息存储类型的定义：

typedef struct node{

char name[8]; /\*选手姓名\*/

int num; /\*选手编号\*/

int score; /\*选手得分\*/

int tax; /\*选手名次\*/

}Node;

## 简单的职工管理系统

任务：对单位的职工进行管理，包括插入、删除、修改、查询、排序等功能。分别使用顺序表、链表、散列表完成存储。将职工对象按散列法存储时，设计解决冲突的方法。并实现以文件方式保存。

要求：职工对象包括姓名、性别、出生年月、工作年月、学历、职务、住址、电话等信息。

（1）新增一名职工：将新增职工对象按姓名以字典方式职工管理文件中。

（2）删除一名职工：从职工管理文件中删除一名职工对象。

（3）查询：从职工管理文件中查询符合某些条件的职工。

（4）修改：检索某个职工对象，对其某些属性进行修改。

（5）排序：按某种需要对职工对象进行排序。

## 全国铁路运输网最佳经由问题

任务：铁路运输网络中有铁路线和火车站的两个主要概念，譬如：1号铁路线表示京广线，2号铁路线表示京沪线等。铁路线对象包括铁路线编号,铁路线名称，起始站编号,终点站编号,该铁路线长度，通行标志(00B客货运禁行,01B货运通行专线，10B客运通行专线，11B客货运通行)。火车站对象包括所属铁路线编号，车站代码，车站名，车站简称，离该铁路线起点站路程及终点站路程。

要求：查询某站所属的铁路线；要求具备新增铁路线的管理功能；要求具备新增车站的管理功能；针对客运，货运情况能计算任何一个起始车站到任何一个终点站之间的最短路径。并且要求能够显示出该最短路径的各个火车站的经由顺序。

## 72教学计划编制问题

任务：大学的每个专业都要制定教学计划。假设任何专业都有固定的学习年限，每学年含两学期，每学期的时间长度和学分上限值均相等，每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的，可以有任意多门，也可以没有。每门课恰好占一个学期。试在这样的前提下设计一个教学计划编制程序。

要求：（1）输入参数包括：学期总数、一学期的学分上限、每门课的课程号(固定占3位的字母数字串)、学分和直接先修课的课程号。（2）允许用户指定下列两种编排策略之一：一是使学生在各学期中的学习负担尽量均匀；二是使课程尽可能地集中在前几个学期中。（3）若根据给定的条件问题无解，则报告适当的信息；否则将教学计划输出到用户指定的文件中。计划的表格格式自行设计。

测试数据：学期总数：6；学分上限：10；该专业共开设12门课，课程号从C01到C12，学分顺序为2,3,4,3,2,3,4,4,7,5,2,3。先修关系如下：

|  |  |  |
| --- | --- | --- |
| 课程编号 | 课程名称 | 先决条件 |
| C1 | 程序设计基础 | 无 |
| C2 | 离散数学 | C1 |
| C3 | 数据结构 | C1，C2 |
| C4 | 汇编语言 | C1 |
| C5 | 语言的设计和分析 | C3，C4 |
| C6 | 计算机原理 | C11 |
| C7 | 编译原理 | C5，C3 |
| C8 | 操作系统 | C3，C6 |
| C9 | 高等数学 | 无 |
| C10 | 线性代数 | C9 |
| C11 | 普通物理 | C9 |
| C12 | 数值分析 | C9，C10，C1 |

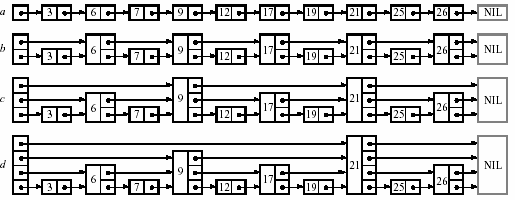
实现提示：可设学期总数不超过12，课程总数不超过100。如果输入的先修课程号不在该专业开设的课程序列中，则作为错误处理。应建立内部课程序号与课程号之间的对应关系。

## 边界标识法

任务：边界标识法是操作系统中用以进行动态分配的一种存储管理的方法，系统将所有的空闲块链接在一个双重循环链表结构中；分配可以按照首次匹配，最佳匹配方法执行。

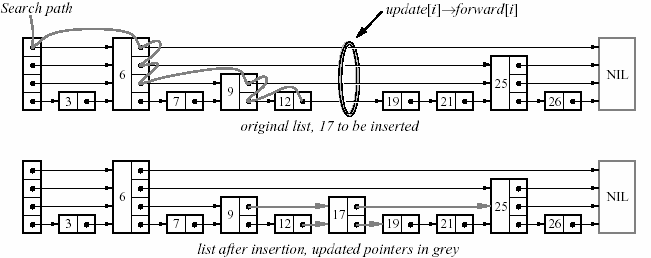
要求：（1）实现分配算法、回收算法（2）显示动态分配回收过程的内存情况。

## Skip List 的实现及分析哈希表设计

任务：Skip List 作为有序链表结构的一种扩展，如下图所示，其中 a 是普通的单链表； 而 b是在此基础上加上第二层（level 2）的额外指针，这些额外的指针指向间隔为 2 的下一个结点，skip list 因此得名；类似的 c 是加上 level 3 后的 skip list；d 是加上 level4 后的 skip list。

Skip List 的基本结构示意图

Skip List 上查找的基本思想是先从最高的 Level 层上查找，找到 key 所在的范围后，再从较低的层次继续重复查找操作，直到 Level 1。Skip List 上的插入操作如下图所示。Skip List 上的删除操作只需直接删除元素即可（包括局部范围内的指针调整），本设计题目的基本内容是构造并实现 Skip List 的 ADT，并能对其维护动态数据集合的效率进行一定的实验验证。

要求：ADT 中应包括初始化、查找、插入、删除等基本操作。分析各基本操作的时间复杂性。针对实现 Skip List 上基本操作的动态演示。能对 Skip List 维护动态数据集合的效率进行实验验证，获得一定量的实验数据，如给定随机产生 1000 个数据并将其初始化为严格 Skip List，在此基础上进行一些列插入、删除、查找操作（操作序列也可以随机生成），获得各种操作的平均时间（或统计其基本操作个数）；获得各操作执行时间的变化情况，应该是越来越大，当大到一定程度后应该进行适当的整理，需设计相应的整理算法，并从数量上确定何时较为合适；能和其他简单线性数据结构，如排序数组上的折半查找进行各类操作效率上的数量对比。

## 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) 采用插入排序、希尔排序、起泡排序、快速排序、选择排序、堆排序、归并排序方法实现上述问题求解。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。

## 药店的药品销售统计系统（排序应用）

任务：设计实现医药公司定期对销售各药品的记录进行统计，可按药品的编号、单价、销售量或销售额做出排名。

要求：在本设计中，首先从数据文件中读出各药品的信息记录，存储在顺序表中。各药品的信息包括：药品编号、药名、药品单价、销出数量、销售额。药品编号共4位，采用字母和数字混合编号，如：A125，前一位为大写字母，后三位为数字，按药品编号进行排序时，可采用基数排序法。对各药品的单价、销售量或销售额进行排序时，采用插入排序、希尔排序、起泡排序、快速排序、选择排序、堆排序、归并排序方法多种排序方法。

药品信息的元素类型定义：

typedef struct node

{

char num[4]; /\*药品编号\*/

char name[10]; /\*药品名称\*/

float price; /\*药品单价\*/

int count; /\*销售数量\*/

float sale; /\*本药品销售额\*/

}DataType;

存储药品信息的顺序表的定义：

typedef struct

{

DataType r[MaxSize];

int length;

}SequenList;

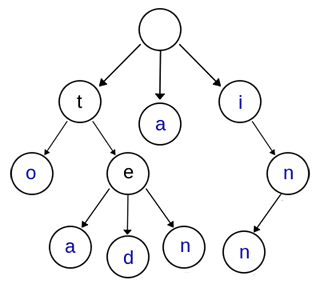
## AVL Tree 的实现及分析

任务：AVL树是平衡的二叉查找树。 编写程序实现 AVL 树的判别；实现 AVL 树的 ADT。

要求：编写 AVL 树判别程序，判别一个二叉查找树是否为 AVL 树。二元查找树用其先序遍历结果表示，如：5，2，1，3，7，8。实现 AVL 树的 ADT；将一般二叉查找树转变为 AVL 树的操作。

## Trie树

任务：**Trie树**，又叫**字典树**、**前缀树（Prefix Tree）**、**单词查找树** 或 **键树**，是一种多叉树结构。如下图：



上图是一棵Trie树，表示了关键字集合{“a”, “to”, “tea”, “ted”, “ten”, “i”, “in”, “inn”} 。从上图可以归纳出Trie树的基本性质：（1）根节点不包含字符，除根节点外的每一个子节点都包含一个字符。（2）从根节点到某一个节点，路径上经过的字符连接起来，为该节点对应的字符串。（3）每个节点的所有子节点包含的字符互不相同。

通常在实现的时候，会在节点结构中设置一个标志，用来标记该结点处是否构成一个单词（关键字）。

可以看出，Trie树的关键字一般都是字符串，而且Trie树把每个关键字保存在一条路径上，而不是一个结点中。另外，两个有公共前缀的关键字，在Trie树中前缀部分的路径相同，所以Trie树又叫做前缀树（Prefix Tree）。

Trie树常被搜索引擎系统用于文本词频统计。为了实现词频统计，我们修改了节点结构，用一个整型变量count来计数。对每一个关键字执行插入操作，若已存在，计数加1，若不存在，插入后count置1。

struct trie\_node

{

int count; // 记录该节点代表的单词的个数

trie\_node \*children[26]; // 各个子节点

};

要求：设计并实现Trie 的 ADT（N=26，建立在英语上的 Trie），该 ADT 包括Trie 的组织存储以及其上的基本操作：包括初始化，查找，插入，删除等；应用 Trie 结构实现文本文档的单词统计。首先扫描文本文档（存放在 txt 文件中），然后在此基础上用 Trie 登记单词出现的次数，最后在 Trie 上实现查询次数。

## 利用Hash技术统计单词的频度

任务：扫描一个C源程序，用Hash表存储该程序中出现的关键字，并统计该程序中的关键字出现的频度。

要求：采用多种方式确定哈希函数，多种方式解决冲突；比较不同算法的效率。

## 通信录查询系统（查找应用）

任务：设计哈希表实现电话号码查询系统。

要求：

1、设每个记录有下列数据项：电话号码、用户名、地址；

2、从键盘输入各记录，分别以电话号码和用户名为关键字建立哈希表；

3、采用多种方式确定哈希函数，多种方式解决冲突；

4、查找并显示给定电话号码的记录；

5、查找并显示给定用户名的记录。

6、考察平均查找长度的变化。

7、通讯录信息文件保存；

8、修改某人的信息，如果未找到要修改的人，则提示通讯录中没有此人的信息。

9、删除某人的信息，如果未找到要删除的人，则提示通讯录中没有此人的信息。

## 导游图

任务：给出一张某公园的导游图，游客从大门进入公园，请帮游客选择一条最佳路线，使他可以不重复地游览各景点，最后回到大门。

要求：将导游图看作一张带权无向图，定点表示公园的各个景点，边表示各景点之间的道路，边上的权值表示距离。

提示：可采用深度优先搜索，如果有多条路径可选择，则选择带权路径最小的路线提供给游客。

## 家族关系查询系统

任务：建立家族关系数据库，实现对家庭成员的相关查询。

要求：（1）建立家族关系并能存储到文件中。（2）实现家族成员的添加。（3）可以查询家族成员的双亲、祖先、兄弟、 孩子和后代等信息。（4）分别采用树状结构和队列。树状结构采用三叉链表表示，队列采用链式队列实现。

## 地铁站建设问题

任务：某城市要在其各个辖区之间修建地铁来加快经济发展，但由于建设地铁的费用昂贵，因此需要合理安排地铁的建设路线，使乘客可以沿地铁到达各个辖区，并使总的建设费用最小。

要求：

（1）从包含各辖区的地图文件中读入辖区名称和各辖区间的直接距离。

（2）根据读入的各辖区的距离信息，计算出应该建设哪些辖区间的地铁路线。

（3）输出应该建设的地铁路线及所需要建设的总里程信息。

## 最短路径：拯救007

任务：看过007系列电影的人们一定很熟悉James Bond这个世界上最著名的特工了。在电影“Live and Let Die”，中James Bond被一组毒品贩子捉住并且关到湖中心的一个小岛上，而湖中有很多凶猛的鳄鱼。这时James Bond做出了最惊心动魄的事情来逃脱——他跳到了最近的鳄鱼的头上，在鳄鱼还没有反应过来的时候，他又跳到了另一只鳄鱼的头上……最后他终于安全地跳到了湖岸上。

假设湖是100×100的正方形，设湖的中心在(0，0)，湖的东北角的坐标是(50，50)。湖  中心的圆形小岛的圆心在(0，0)，直径是15。一些凶残的鳄鱼分布在湖中不同的位置。现已知湖中鳄鱼的位置(坐标)和James Bond可以跳的最大距离，请你告诉James Bond一条最短的到达湖边的路径。他逃出去的路径的长度等于他跳的次数。

要求：程序从“input.txt”，文件中读取输入信息，这个文件包含了多组输入数据。每组输入数 据的起始行中包含两个整数n和d，n是鳄鱼的数量而且n≤100，d是007可以跳的最大距离而且d>0。起始行下面的每一行是鳄鱼的坐标(x，y)，其中x，y都是整数，而且没有任何两只鳄鱼出现在同一个位置。input.txt文件以一个负数结尾。 输出：

程序结果输出到output.txt文件中。对于每组输人数据，如果007可以逃脱，则输出到output.txt文件的内容格式如下：第一行是007必须跳的最小的步数，然后下面按照跳出顺序记录跳出路径上的鳄鱼坐标(x，y)，每行一个坐标。如果007不可能跳出去，则将-1写人文件。如果这里有很多个最短的路径，只需输出其中的任意一种。

输入例子：

4  10            /\*第一组输人数据\*/

17  0

27  0

37  0

45  0

1  10          /\*第二组输入数据\*/

20  30

-1

输出例子：

5              /\*对应第一组数据的输出\*/

17  0

27  0

37  0

45  0

-1              /\*对应第二组数据的输出\*/

提示：将每个鳄鱼看作图中的一个顶点。如果007可以从A点跳到B点，则A和B之间就有一条边。

进阶：用GUI的形式实现。

## 图书管理系统（查找应用）

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：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。由于图书管理的基本业务活动都是通过书号（即关键字）进行的，所以要用对书号索引，以获得高效率。系统应实现的基本功能有：采编入库：新购入一种书，经分类和确定书号之后登记到图书帐目中去。如果这两种书在帐中已有，则只将总库存量增加。清除库存：某种书已无保留价值，将它从图书帐目中注销。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。归还：注销对借阅者的登记，改变该书的现存量。

## 四则运算表达式

任务：键盘输入一个含有括号的四则运算表达式，可能含有多余的括号，编程整理该表达式，要求：去掉所有多余的括号，原表达式中所有变量和运算符相对位置保持不变，并保持与原表达式等价。并对表达式求值。

## 猴子选大王

任务：一堆猴子都有编号，编号是1，2，3 ...m ,这群猴子（m个）按照1-m的顺序围坐一圈，从第1开始数，每数到第N个，该猴子就要离开此圈，这样依次下来，直到圈中只剩下最后一只猴子，则该猴子为大王。

要求：至少采用两种不同的数据结构的方法实现。

## 24点游戏

任务：由系统随机生成4张牌，用户利用扑克牌的数字及运算符号“+”、“—”、“\*”、“/”及括号“（”和“）”从键盘上输入一个计算表达式，系统运行后判断是否为24。

要求：（1）用户输入求解（2）自动求解

## 背包问题的求解

任务：假设有一个能装入总体积为T的背包和n件体积分别为w1 , w2 , … , wn 的物品，能否从n件物品中挑选若干件恰好装满背包，即使w1 +w2 + … + wn=T，要求找出所有满足上述条件的解。例如：当T=10，各件物品的体积{1，8，4，3，5，2}时，可找到下列4组解：（1，4，3，2）（1，4，5）（8，2）（3，5，2）。

要求：可利用回溯法的设计思想来解决背包问题。首先将物品排成一列，然后顺序选取物品装入背包，假设已选取了前i 件物品之后背包还没有装满，则继续选取第i+1件物品，若该件物品"太大"不能装入，则弃之而继续选取下一件，直至背包装满为止。但如果在剩余的物品中找不到合适的物品以填满背包，则说明"刚刚"装入背包的那件物品"不合适"，应将它取出"弃之一边"，继续再从"它之后"的物品中选取，如此重复，，直至求得满足条件的解，或者无解。由于回溯求解的规则规则是"后进先出"因此自然要用到栈。

## 扫雷

任务：做一个N x M的扫雷游戏，每个方格包含两种状态：关闭（closed）和打开（opened），初始化时每个方格都是关闭的，一个打开的方格也会 包含两种状态：一个数字（clue）和一个雷（bomb）。你可以打开（open）一个方格，如果你打开的是 一个bomb，那么就失败；否则就会打开一个数字，该数字是位于[0，8]的一个整数，该数字表示其所有邻居方格（neighboring squares）所包含的雷数。

要求：1.能够打开一个方格，一个已打开的方格不能再关闭。2.能够标记一个方格，标记方格的含义是对该方格有雷的预测（并不表示真的一定有雷），当一个方格标记后该方格不能被打开，只能执行取消 标记的操作，只能在取消后才能打开一个方格。3.能够给出游戏结果（输、赢、剩余的雷数、用掉的时间按秒计）。

## 简单行编辑程序。

任务：文本编辑程序是利用计算机进行文字加工的基本软件工具，实现对文本文件的插入、删除等修改操作。限制这些操作以行为单位进行的编辑程序称为行编辑程序。

被编辑的文本文件可能很大，全部读入编辑程序的数据空间（内存）的做法既不经济，也不总能实现。一种解决方法是逐段地编辑。任何时刻只把待编辑文件的一段放在内存，称为活区。试按照这种方法实现一个简单的行编辑程序。设文件每行不超过320个字符，很少超过80字符。

要求：实现以下4条基本编辑命令：

（1） 行插入。格式：i<行号><回车><文本><回车>

将<文本>插入活区中第<行号>行之后

（2）行删除。格式：d<行号1>[□<行号2>]<回车>

　　删除活区中第<行号1>行（到第<行号2>行）。两种格式的例子是：“d10↙”和“d10□14↙”

（3）活区切换。格式：n<回车>

将活区写入输出文件，并从输入文件中读入下一段，作为新的活区。

（4）活区显示。格式：p<回车>

逐页地（每页20行）显示活区内容，每显示一页之后请用户决定是否继续显示以后各页（如果存在）。印出的每一行要前置以行号和一个空格符，行号固定占4位，增量为1。

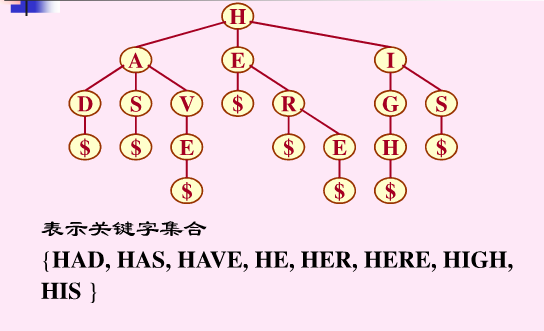
各条命令中的行号均须在活区中各行行号范围之内，只有插入命令的行号可以等于活区第一行行号减1，表示插入当前屏幕中第一行之前，否则命令参数非法。

提示：（1） 设活区的大小用行数activemaxlen（可设为100）来描述。考虑到文本文件行长通常为正态分布，且峰值在60到70之间，用320×activemaxlen大小的字符数组实现存储将造成大量浪费。可以以标准行块为单位为各行分配存储，每个标准行块含81个字符。这些行块可以组成一个数组，也可以利用动态链表连接起来。一行文字可能占多个行块。行尾可用一个特殊的ASCII字符（如(012)8）标识。此外，还应记住活区起始行号。行插入将引起随后各行行号的顺序下推。（2） 初始化过程包括：请用户提供输入文件名（空串表示无输入文件）和输出文件名，两者不能相同。然后尽可能多地从输入文件中读入各行，但不超过activemaxlen-x。x的值可以自定，例如20。（3） 在执行行插入命令的过程中，每接收到一行时到要检查活区大小是否已达activemaxlen。如果是，则为了在插入这一行之后仍保持活区大小不超过activemaxlen，应将插入点之前的活区部分中第一行输出到输出文件中；若插入点为第一行之前，则只得将新插入的这一行输出。（4） 若输入文件尚未读完，活区切换命令可将原活区中最后几行留在活区顶部，以保持阅读连续性；否则，它意味着结束编辑或开始编辑另一个文件。（5） 可令前三条命令执行后自动调用活区显示。

选作内容：（1） 对于命令格式非法等一切错误作严格检查和适当处理。（2） 加入更复杂的编辑操作，如对某行进行串替换；在活区内进行模式匹配等，格式可以为S<行号>@<串1>@<串2><回车>和m<串><回车>。

## 双链树

任务：键树又称数字查找树（Digital Search Tree）。它是一棵度大于等于2的树，树中的每个结点中不是包含一个或几个关键字，而是只含有组成关键字的符号。例如，若关键字是数值，则结点中只包含一个数位；若关键字是单词，则结点中只包含一个字母字符。这种树会给某种类型关键字的表的查找带来方便。如下图所示为一棵键树：



　　从根到叶子结点路径中结点的字符组成的字符串表示一个关键字，叶子结点中的特殊符号$表示字符串的结束。在叶子结点中还含有指向该关键字记录的指针。通常，键树可有两种存储结构，分别称为**双链树**和**Trie树**。

双链树，以树的**孩子兄弟链表**来表示键树，则每个分支结点包括三个域：

　　symbol域：存储关键字的一个字符；

　　first域：存储指向第一棵子树根的指针；

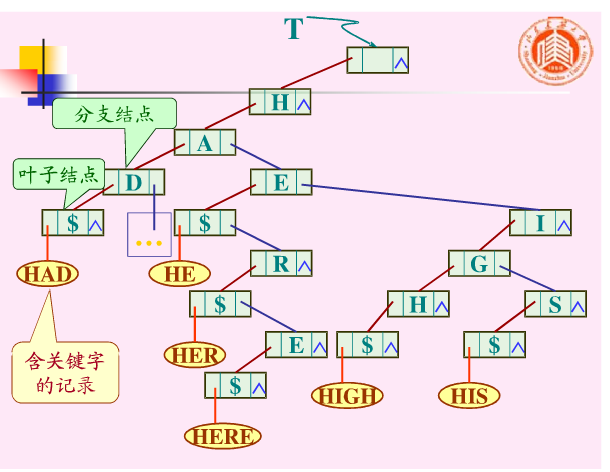
　　next域：存储指向右兄弟的指针。

　　同时，叶子结点不含first域，它的infoptr域存储指向该关键字记录的指针。

　　在双链树中插入或删除一个关键字，相当于在树中某个结点上插入或删除一棵子树。

　　结点的结构中可以设置一个枚举变量表示结点的类型，叶子结点和分支结点。

　　叶子结点和分支结点都有symbol域和next域。不同的一个域可以用联合表示，叶子结点包含infoptr指向记录，而分支结点是first域指向其第一棵子树。双链树如下图：



 要求：设计并实现双链树的 ADT（关键字是单词），该 ADT 包括双链树的组织存储以及其上的基本操作：包括初始化，查找，插入，删除等；应用双链树结构实现文本文档的单词统计。首先扫描文本文档（存放在 txt 文件中），然后在此基础上登记单词出现的次数，最后实现查询次数。

## 二叉树的叶子结点按从左到右的顺序连成一个单链表

任务：设计一个算法，把二叉树的叶子结点按从左到右的顺序连成一个单链表。

要求：二叉树用二叉链存储，链接时用叶子结点的rchild 域存放指针。

## 找出X和Y的一个最长公共子序列

任务：给定两个序列X=<x1, x2, …, xm>和Y=<y1, y2, … , yn>，要求找出X和Y的一个最长公共子序列。

要求：至少采用两种不同的数据结构的方法实现。

## 分子量

任务：分子式是用来表达分子组成结构的表达式，一般表达形式为A1c1A2c2A3c3... 其中Ai(i=1,2,.......)表示原子或原子团，ci(i=1,2,....)表示原子或原子团Ai重复的次数。当ci=1时，ci必须省略不写，且原子团的括号也不要。例如N的原子量为14，H的原子量为1，C的原子量为12，O的原子量为16，因此(NH4）2CO3的分子量为（14+1\*4）\*2+12+16\*3=96。试编写程序求出给定的各个分子式所对应的分子量。

要求：能对错误的表达式进行提示。能求出分子量。

## 公交线路上优化路径的查询

任务：最短路径问题是图论中的一个经典问题，其中的 Dijkstra 算法一直被认为是图论中的好算法，但有的时候需要适当的调整 Dijkstra 算法才能完成多种不同的优化路径的查询。对于某城市的公交线路，乘坐公交的顾客希望在这样的线路上实现各种优化路径的查询。设该城市的公交线路的输入格式为：

线路编号：起始站名(该站坐标)；经过的站点1名(该站坐标)；经过的站点2名(该站坐标)；……；经过的站点n名(该站坐标)；终点站名(该站坐标)。该线路的乘坐价钱。该线路平均经过多少时间来一辆。车速。

例如：63：A(32,45)；B(76,45)；C(76,90)；……；N(100,100)。1元。5分钟。1/每分钟。30KM/H

假定线路的乘坐价钱与乘坐站数无关，假定不考虑公交线路在路上的交通堵塞。

任务：对这样的公交线路，需要在其上进行的优化路径查询包括：任何两个站点之间最便宜的路径；任何两个站点之间最省时间的路径等等。

根据上述公交线路的输入格式，定义并建立合适的图模型。

针对上述公交线路，能查询获得任何两个站点之间最便宜的路径，即输入站名 S，T 后，可以输出从 S 到 T 的最便宜的路径，输出格式为：线路 x：站名 S，…，站名 M1； 换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 元。 针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（不考虑在中间站等下一辆线路的等待时间），即输入站名 S，T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x：站名 S，…，站名 M1；换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 时间。针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（要考虑在中间站等下一辆线路的等待时间），即输入站名 S，T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x：站名 S，…，站名 M1； 换乘线路 x：站名 M1，…，站名 M2；…；换乘线路 x：站名 MK，…，站名 T。共花费 x 时间。

## CD 数据管理

任务：完成存储歌曲，每首歌曲包含三个属性：歌曲作者、歌曲题目和歌曲创作的年份，即(Artist, Title, year)。数据中的歌曲被存放在一个数据存储区中，且这些歌曲按照(Artist, Title, year)这三个属性组织成三个有序链表，目的是提高诸如 getByYearRange 等算法的效率。

要求：应该提供如下基本操作：addSong(Song a); removeSong(Song a);getByArtist(String artist); getByTitle(String title); getByYear(int year); getByArtistRange(String start, String finish); getByTitleRange(String start, String finish); getByYearRange(int start, int finish); printByArtist(PrintWriter out); printByTitle(PrintWriter out); printByYear(PrintWriter out);

实现 Playlist 操作。addSongToPlaylist(Song s); removeSongFromPlaylist(Song s);getNextSong(); length();isEmpty(); shufflePlaylist();//搅乱(洗牌)playlist 中的歌曲，自己设计一个搅乱算法。playlist 被组织成一个指向歌曲对象“指针”(references to song objects)的数组。

将上部分的 playlist 扩展成一个 sortable playlist，该 sortable playlist 支持以下操作sortByArtist(); sortByTitle();sortByYear();sortable playlist 中采用的 sort 方法是自底向上的 mergesort 算法。

在上述操作上编写一个小型应用程序，该程序能够测试(演示)上述操作，同时该应用程序还应完成诸如存入磁盘、从磁盘载入等功能。

## 模拟理发店并对其进行数据分析

任务：一个理发店是这样的一个系统：理发店内有 m 个理发椅，每个理发椅对应一个理发师，即有 m 个理发师，理发店内有 n 个等候座位。在理发店内添置一个理发椅(同时雇佣一个理发师)的代价为 C1，在理发店内添置一个等候座位的代价为 C2，由于物理空间和理发店的经济条件的限制，要求 mC1+nC2CT。假定理发店系统的工作过程如下：当一个顾客进入理发店时，如果有空闲的理发师(理发椅)时，顾客就坐在该理发椅上进行理发，顾客的理发时间 T1 符合指数分布 E1，顾客的理发费用，即理发店可以获得的收费为 cT1，其中 c 为每个单位时间的收费；如果没有空闲的理发师(理发椅)时，顾客就看有没有空闲的等候座位，如果有就坐在一个等候座位上，顾客坐在等候座位上的等候时间 T2 满足指数分布 E2，如果在该段时间内出现了空闲的理发椅并轮到了自己，顾客就去理发，否则顾客就离去。如果没有等候座位，顾客就站在店里或站在门外等候(这就表示站着的顾客个数是不受限制的)，顾客站着等候的时间 T3 满足指数分布 E3，如果在该段时间内有了空闲的等候座位并轮到了自己，顾客就在座位上继续等待，坐在座位上再等的时间 T4 和用户已经站着等待的时间 T5 之和 T4+T5 满足指数分布 E4，坐在座位上的顾客的工作方式同前，如果等待的时间用完，用户就离去。用户在理发店的营业时间内按泊松分布 B 到达理发店。完成上述理发店系统的模拟，在完成该模拟时需要：仿照实际情况对上述四个指数分布 E1 到 E4 和泊松分布 B 的基本参数进行设定；仿照实际情况对 c，C1，C2 和 CT 进行设定；设定模拟时间，设定 m 和 n 等参数。学会完成计算机对一些常见的复杂系统进行模拟，学会对一些常见的数学分布进行计算模拟，学会应用线性数据结构（尤其是队列）进行存储和组织信息。

要求:① 实现理发店系统的模拟。② 根据模拟结果得出不同的 m 和 n 组合获得理发店收益。③ 模拟系统应该有较为完整地，能有效说明模拟过程的界面。④ 模拟系统的配置参数应该很容易修改(如采用配置文件或在图形界面中用控件输入)。⑤ 如果可以的话，给出理发店系统更为合理的模拟，即考虑更多的模拟细节。

## 求出m×n矩阵的所有马鞍点。

任务：矩阵A中的元素若满足：A[i,j]是第i行中值最小的元素，且又是第j列中值最大的元素，则称元素A[i,j]为该矩阵的一个马鞍点。求出m×n矩阵的所有马鞍点。

要求：使用二维数组、堆分配数组(教材P93)、三元组、十字链表完成上述操作并比较效率。

## B树，B+树的应用

任务：B+Tree是在B-Tree基础上的一种优化，使其更适合实现外存储索引结构，InnoDB存储引擎就是用B+Tree实现其索引结构。查阅相关资料，例如<https://www.cnblogs.com/hdk1993/p/5840599.html>，编写B+Tree外存储索引结构仿真系统。

要求：完成索引的建立和查找。

## 找到100亿个URL中重复的URL

题目：有一个包含100亿个URL的大文件，假设每个URL占用64B，请找出所有重复的URL。

要求：内存限制为2GB。

## 每天最热100词的可行办法

题目：某搜索公司一天的用户搜索词汇是海量的，假设有百亿的数据量，请设计一种求出每天最热100词的可行办法

要求：内存限制为2GB。

思路：还是用hash函数分流，然后在用hash表做词频统计，如果机器数有限，则需要分流每台机器上的文件，再用hash表做词频统计。

创建hash表后，可以利用小根堆来进行top100筛选，得到每个小文件排序后的top100，然后再将同一台机器上所有文件的top100通过小根堆或外排得到每台机器上的排序后的top100，最后再通过小根堆或外排得到所有机器也就是所有数据桶的排序后的top100，返回即可。