2019级“数 据 结 构 与 程 序 设 计”期末考试

《数据结构与程序设计》期末考试须知

1. 请按照统一提前下发的“《数据结构与程序设计》期末考试方案”中的相关要求设置考试环境!
2. 将在考试正式开始前十分钟下发试卷!
3. 请使用Google Chrome，360极速或Firefox浏览器登录考试平台。
4. 开卷考试，考生可以查阅纸质资料、考试机器中的电子资料(PPT，数据结构教程(唐发根编著)和C程序设计导引(尹宝林 编著))等，但不能上网查资料(包括邮箱、即时通讯软件等)，不能通过网络与他人进行任何形式的交流。
5. 编程题提交后，将会返回运行结果，即:测试数据是否正确的信息，但不会返回详细评判信息。注意:当同一时间提交的程序文件较多时，尤其是考试最后几分钟服务器繁忙时，成功提交的反馈信息为:“已成功提交”，评判结果信息的反馈可能有延迟。 同学们提交程序后注意查看反馈信息，确认程序能够成功提交!
6. 程序文件可以多次提交，以最后一次提交为准。
7. 程序中输入/输出必须按照程序要求(必须和输入/输出样例例完全相同)，不要填加任何额外信息，否则得分为0，输出数据的最后有没有回车换行行都可以。
8. 在题目中没有特殊要求情况下:填空题答案中不得有空格，选择题答案只填大写或小写字母，不得有括号等其它字符!
9. 考试将全程监控，未经监考老师同意，不得自行更换机器，一个帐号在多个机器上登录，或者一个机器上登录多个帐号(即: 以其他同学帐号登录)，都以作弊处理。
10. 提前交卷的同学，请点击考试页面右上角的“交卷”按钮，在考试结束前，不得再次登录考试系统，否则以作弊处理。提前交卷的同学可以离开机位，但在考试结束前不得退出腾讯会议，不得改动监控摄像头位置，不得移动电脑!16：30分之后提前交卷的同学不得离开座位!
11. 对于不能直接用快捷键拷贝贴的Console窗口，为了提高调试程序时输入数据的效率，可以先复制待输入的数据，然后右键单击Win32 Console窗口最上方方的标题栏，选择“编辑”菜单项中的“粘贴”命令即可。
12. 用用Dev-C++调试时，编写的程序不能放在硬盘的根目录下，请自建子目录!目录名不建议用中文!
13. 考试时间到，应及时点击试题右上方的“交卷”按钮并退出考试平台。若无故不交卷，出现问题学生自己负责。在提交程序或若交卷过程中出现问题，请通过微信或电话及时联系监考老师。
14. 考试期间，若网络正常情况下出现**代码无法提交**的情况，重新刷新浏览器，或者关闭浏览器，重新登录judge平台。

选择题(共15道，每道1分)

1. 【单选题】

下面程序段的时间复杂度为　　。

```
c=0;
 for(i=0; i<m; i++)
    for(j=0; j<n; j++ )
       c=c+i*j;
```

A.O(m2) B.O(n2) C.O(m×n) D.O(m+n)

2.【单选题】

设某一非空的单循环链表的头指针(指向第一个结点的指针)为Head，尾指针(指向最后一个结点的指针)为Rear，则下列条件判断结果一定为真的是

A. Head== Rear->link B. Head== Rear->link->link C. Rear== Head->link D. Rear== Head->link->link

3.【单选题】

若5个元素的出栈序列为1，2，3，4，5，则进栈序列可为　　。

A.2,4,3,1,5 B.2,3,1,5,4 C.3,1,4,2,5 D.3,1,2,5,4

4. 【单选题】

设栈S和队列Q的初始状态均为空，元素a,b,c,d,e, f,依次进入栈S。若每个元素出栈后立即进入队列Q, 且6个元素出队的顺序是b, f ,e , d, c,a，则栈S的容量至少是 。

A.5 B.3 C.2 D.6

5. 【单选题】

下列语句中，能正确进行字符串赋值的是 。
A. char* sp; *sp="BUAA" ; B. char s[10]; s="BUAA";
C. char s[10]; *s="BUAA"; D. char *sp="BUAA";

6.【单选题】

下列说法中，错误的是
A. 在图的邻接矩阵存储中，无向图的邻接矩阵一定是一个对称矩阵;
B. 在图的邻接表存储中，有向图的第i个链表中边结点个数是第i个顶点的出度;
C. 包含具有n个顶点的连通图G的全部n个顶点,仅包含其n-1条边的极小连通子图称为G的一个生成树;
D. 对于给定的带权连通无向图，从某源点到图中各顶点的最短路径构成的生成树一定是该图的最小生成树;

7.【单选题】

可用于求无向图的所有连通分量的算法是

A.广度优先遍历 B.拓扑排序 C.求最短路径 D.求关键路径

8. 【单选题】

设有一组记录的关键字为{19，16，23，1，68，20，84，27，55，11，10，75}，用链地址法构造散列表，散列函数为H(key)=key MOD 13，散列地址为1的链中有　　个结点。
A.1 B. 2 C. 3 D. 4

9.【单选题】

若在有序序列中采用折半查找方法进行查找，用来描述该查找过程的“判定树”的形状与　　有关。

A.序列中元素的值 B.序列中元素的排列次序 C.序列中元素的类型 D.序列中元素的个数

10. 【单选题】

若用一个大小为6的数组来实现循环队列，且当前rear和front的值分别为0和3，当从队列中出队一个元素，再入队两个元素后，rear和front的值分别为 。

A. 1和5 B. 2和4 C. 4和2 D. 5和1

11.【单选题】

一个具有767个结点的完全二叉树，其叶子结点个数为　　。
A. 383 B. 384 C. 385 D. 386

12. 【单选题】

在二叉排序树中进行查找的效率与

A. 二叉排序树的深度
B. 二叉排序树的叶结点的个数
C. 被查找结点的度
D. 二叉排序树的存储结构

13. 【单选题】

在下面的查找方法中， 不是基于关键字值比较的查找方法。

A.顺序查找法 B.折半查找法 C.索引查找法 D.散列查找法

14. 【单选题】

下列排序方法中，不稳定的排序方法是

A.冒泡排序 B. 快速排序 C.归并排序 D. 插入排序

15. 【单选题】

对一组数据(84，47，25，15，21)排序，排序过程如下:

(1) 15 47 25 84 21
(2) 15 21 25 84 47
(3) 15 21 25 84 47
(4) 15 21 25 47 84

则采用的排序算法是 。

A. 选择 B. 冒泡 C. 快速D. 插入

填空题(5道，每道2分)

1. 若某完全二叉树采用顺序存储结构，结点存放的次序为ABCDEFGH，请给出该二叉树的后序序列(字符之间不加任何分隔符号) 。
2. 假设采用快速排序算法按照从小到大的顺序对10个数据元素进行排序，这10个数据的初始状态为(57,11,12,-23,76,1901,20,96,-38,0)，若第一次选用第一个数据57作为分 界元素，则第一趟排序后，分界元素57前有　　个数据元素。
3. 有9个叶结点的哈夫曼树共有　　个结点。
4. n个科技园区，现要为它们建成能相互通讯的有线网，并且使线路长度总和最短，这可以归结为图的　　的问题。
5. 若某栈初始为空，符号PUSH与POP分别表示对栈进行1次进栈操作与1次出栈操作，那么，对于输入序列ABCDE，经过PUSH, PUSH, POP, PUSH, POP, PUSH, PU SH, POP以后，出栈序列应该是(字符之间不加任何分隔符号)。

编程题

1. 空 闲 空 间 合 并(25分)

【问题描述】

一个内存空间块可用一对起始地址和结束地址表示，如[0,100]。对于两个内存空间块：[0,100]和[101,200]，这两个空间是相邻的，所以可以合并成一个空间块：[0,200]。编写程序，输入一组空闲内存空间，对相邻空间进行合并，然后按照空间块的起始地址由小到大输出。注意：内存空间块不会存在重叠的情况。

算法之一提示：可先将输入的空间块按起始地址排序，然后进行合并。

【输入形式】

从控制台输入内存空间块的个数（大于等于1，小于等于100），然后分行输入每个内存空间块的起始地址和结束地址（地址用大于等于0，并且小于等于100000的整数表示），两地址间以一个空格分隔。

【输出形式】

将合并后的内存空间块按照空间块的起始地址由小到大输出，每个空间块独占一行，先输出起始地址，再输出结束地址，两地址间以一个空格分隔。

【样例输入】

10

48 99

0 39

1024 2047

100 479

4000 5999

600 799

40 47

2048 3047

840 859

8000 8999

【样例输出】

0 479

600 799

840 859

1024 3047

4000 5999

8000 8999

【样例说明】

输入了10个内存空间块，其中：[0,39]、[40,47]、[48,99]和[100,479]相邻，可以合并成一个空间块：[0,479]；[1024,2047]和[2048,3047]相邻，可以合并成[1024,3047]。合并后还有6个内存空间块，按照空间块的起始地址由小到大输出。

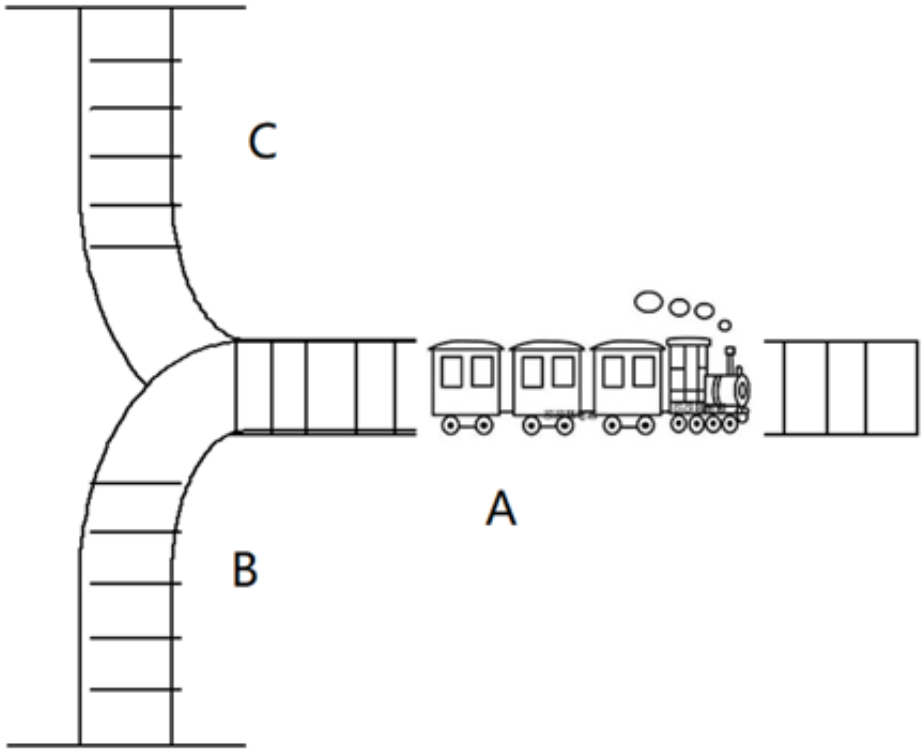
【评分标准】

该题要求输出合并后的内存空间，提交程序名为space.c。

2. 火车货运调度模拟(20分)

【问题描述】

某火车货场由A、B、C三段组成，见下图。现有一列货车停在A段上，由多个货物车厢组成，每节车厢信息包括编号和货物发往的目的地（车厢编号是唯一的，各节车厢发往的目的地可以相同，也可以不同）。当前停在A段的货车车厢发往目的地编组是乱的，需要按目的地由远至近进行重新编组，即车厢离车头越近其目的地越远，这样方便到达目的地时卸货（卸下相关车厢）。编组过程中车厢只能依次从A中移动至B或C中，或从B或C中移至A中，从B中不能直接移至C中，从C中也不能直接移至B中。编写一程序模拟货运车厢编组。



假设A、B、C三段交叉路口为各段的顶端，编组规则如下：

1. 初始时所有车厢位于A中且均未完成编组，车头距离顶端最远；首先将其所有车厢从顶开始依次推进至B中。
2. 从B中找到发往地最远的车厢中离顶最近的车厢，假设其为M。将M至顶的所车厢依次推进至A中，此时M一定位于A中顶端。
3. 若A中M之后（即：M与车头之间）不存在未完成编组的车厢，则车厢M的编组完成了；若A中M之后存在未完成编组的车厢，则将M推进至 C中，将A中M之后所有未完成编组的车厢依次推进至B中，再将M由C中推进至A中，这样就完成了车厢M的编组；
4. 重复步骤2-3，直到B中无车厢。

【输入形式】

先从控制台输入目的地个数（大于等于1，小于等于50），然后分行输入一组由地名（用不含空白符的英文字符串表示，字符个数不超过20）和里程（用大于等于1小于等于10000的整数表示）构成的发往目的地（由近至远序）里程表，地名和里程之间以一个空格分隔；在里程表之后输入车厢个数（大于等于1，小于等于50），然后分行输入一组由车厢编号（由四位数字组成）和发往目的地（目的地肯定在上述里程表中出现）构成的车厢信息，车厢编号和目的地之间以一个空格分隔，并且是从车头处开始依次输入每节车厢信息。

【输出形式】

假设一节车厢从某段推出称为一次pop操作，推进某段称为一次push操作。程序运行输出第一行为从车头开始编好组的车厢编号，中间用一个空格隔开，最后一个车厢编号后也有一个空格。第二行输出编组过程中A段中push操作的次数。

【样例输入】

10

shijiazhuang 280

xingtai 390

xinxiang 610

zhengzhou 689

wuchang 1221

chibi 1339

yueyang 1434

changsha 1559

shaoguan 2057

guangzhou 2273

12

0039 guangzhou

5217 xingtai

0262 yueyang

7205 wuchang

3211 guangzhou

4893 shijiazhuang

2283 shaoguan

0890 guangzhou

8729 wuchang

6839 shijiazhuang

2122 changsha

3280 wuchang

【样例输出】

0039 3211 0890 2283 2122 0262 7205 8729 3280 5217 4893 6839

45

【样例说明】

首先输入由10个地名及其里程组成的里程表，然后输入了12节需要编组的车厢的编号和发往的目的地，即初始时处于A中的车厢，其中0039编号的车厢离车头最近。按照上面编组规则，首先将所有车厢推进至B中，这时0039车厢位于B的顶端，3280车厢位于底端，B中所有车厢中最远目的地为guangzhou、且离顶最近的车厢为0039，将其推进至A中，这时A中只有其一节车厢，其后没有未编组的车厢，0039车厢编组完成（即完成第一节车厢编组），此时A的push操作次数为1；接下来，B中剩余的11个车厢中，最远目的地依然为guangzhou，其离顶最近的车厢为3211，将该车厢及其上的3节车厢依次推进到A中，此时，3211车厢与火车头之间有三节车厢未完成编组，于是按规则将3211推进至C中，将7205、0262和5217依次推进至B中，再将C中的3211推进至A中，这时3211车厢编组完成（即完成第二节车厢编组），此时A的push操作次数为6；再依次按照上述步骤对B中剩余的车厢完成编组，直到B中无车厢。最后从车头开始将所有完成编组的车厢编号依次输出，由此得到一组发往目的地距离（从车头开始）由远至近的车厢序列。完成编组过程中A的push操作次数共有45次。

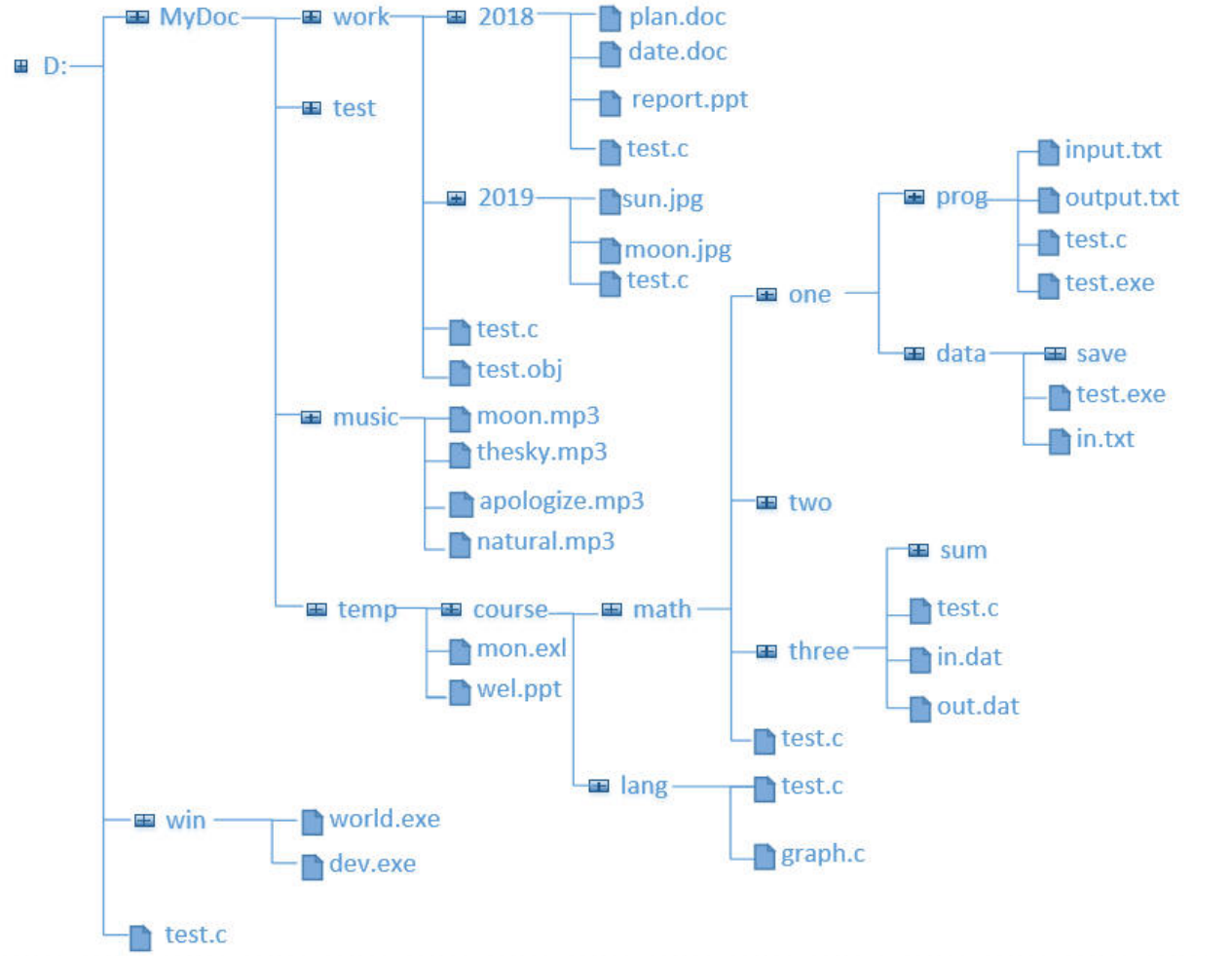
【评分标准】

该题要求编程模拟火车车厢编组操作，提交程序名为：train.c。

3. 查找同名文件(10分)

【问题描述】

在操作系统中目录及文件是按树形式组织和管理的，从根目录开始，每一层目录下可以包含子目录和文件。假设一文件系统，从根目录开始，每个目录下可包含最多不超过100个文件及子目录，文件名及目录名是一字符串（不包含空白符，最多包含20个字符），每个文件和目录包含修改时间属性（仅由日期组成的字符串，例如：20190927表示2019年9月27日），在同一个目录下不会存在同名文件，而且所有的目录都不会重名。例如，下图是一给定的文件结构示意：



给定一文件名，请在给定文件结构中查找同名文件，并按修改时间由近至远排列输出其包含全路径名的文件名，即从根目录开始的目录及文件名（包含的字符个数不会超过200）；修改时间相同时，从根目录开始按层次由小到大的顺序输出；修改时间和层次都相同的，按照由上到下的顺序输出。假设找到的同名文件个数不会超过100。

【输入形式】

从控制台输入文件结构中目录及文件的总个数和待查找的文件名，中间由一个空格分隔；然后从当前目录下文件files.txt中分行读入每个目录及文件的信息，其格式如下：

name parentName type date

name是一字符串，表示目录或文件的名字，其中不包含空白符，最多包含20个字符；parentName表示该目录或文件的上级目录名字，根目录没有上级目录，用字符"-”表示；type为0或1，用于区分当前输入的是文件还是目录：0表示文件，1表示目录；date为一日期字符串，表示目录或文件的修改时间，例如：20190927表示2019年9月27日。

读入的目录或文件遵循如下原则：读入当前目录或文件时，其上级目录已经读入；对于同一目录下的文件或子目录，先读入的位于前面（显示在上方）。

【输出形式】

按修改时间由近至远排列分行输出找到的文件：包括其从根目录开始的各级目录及文件名，目录及文件之间以一个英文字符“"分隔；修改时间相同时，按从根目录开始按层次由小到大的顺序输出；修改时间和层次都相同的，按照由上到下的顺序输出。注意：由于根目录一般为盘符，输出时要加英文字符":"，例如若根目录名为D，则输出时应为 D:。

【样例输入】

49 test.c

当前目录下files.txt文件中内容如下：

D - 1 20190101

MyDoc D 1 20190101

win D 1 20190101

test.c D 0 20190901

work MyDoc 1 20190101

2018 work 1 20190102

plan.doc 2018 0 20190202

date.doc 2018 0 20190305

2019 work 1 20190103

test.c work 0 20190901

test.obj work 0 20190902

sun.jpg 2019 0 20190405

report.ppt 2018 0 20190306

test MyDoc 1 20190103

music MyDoc 1 20190103

temp MyDoc 1 20190103

moon.mp3 music 0 20190607

thesky.mp3 music 0 20190701

moon.jpg 2019 0 20190506

test.c 2019 0 20190901

apologize.mp3 music 0 20190702

test.c 2018 0 20190901

natural.mp3 music 0 20190702

world.exe win 0 20190101

dev.exe win 0 20190102

course temp 1 20190505

mon.exl temp 0 20190505

wel.ppt temp 0 20190605

math course 1 20190506

lang course 1 20190607

one math 1 20190507

test.c lang 0 20190925

graph.c lang 0 20190926

two math 1 20190508

three math 1 20190508

test.c math 0 20190926

prog one 1 20190609

data one 1 20190610

sum three 1 20190509

test.c three 0 20190901

in.dat three 0 20190902

input.txt prog 0 20190809

output.txt prog 0 20190827

test.c prog 0 20190901

test.exe prog 0 20190901

save data 1 20190611

test.exe data 0 20190612

in.txt data 0 20190612

out.dat three 0 20190902

【样例输出】

D:\MyDoc\temp\course\math\test.c

D:\MyDoc\temp\course\lang\test.c

D:\test.c

D:\MyDoc\work\test.c

D:\MyDoc\work\2018\test.c

D:\MyDoc\work\2019\test.c

D:\MyDoc\temp\course\math\three\test.c

D:\MyDoc\temp\course\math\one\prog\test.c

【样例说明】

files.txt中有49个目录和文件，形成如上图所示的文件系统。待查找的文件名为test.c。可以在该文件系统中找到8个名为test.c的文件，其中D:\MyDoc\temp\course\math目录下的test.c最新，所以先输出，然后是D:\MyDoc\temp\course\lang目录下的test.c文件。剩下六个test.c文件的修改日期相同，按照从根目录D:开始层次由小到大顺序输出。其中：目录D:\MyDoc\work\2018和D:\MyDoc\work\2019下的test.c文件所处层次相同，则按照由上至下的顺序输出。

【评分标准】

该题要求查找文件所在的目录，提交文件名为find.c。

选择填空参考答案

同学做的版本，仅供参考。

选择题

1. `C`
   双重循环，时间复杂度为相乘。
2. `A`
   head = tail->next;
3. `D`
   A不行，2 4 3 1入栈后出栈1， 2出不来。
   B不行，2 3 1入栈后出1, 2出不来。
   C不行，3 1入，1出，4 2入(此时栈为[3 4 2])，2出，此时3出不来。
   D可以，3入，1入，1出，2入，2出，3出，5入，4入，4出，5出。
4. `A`
   a入，b入，b出，c入，d入，e入，f入, f出, e出，d出，c出，a出。
   [a], [ab], [a], [ac], [acd], [acde], [acdef], [acde], ...
5. `D`
   A不行，`*sp=`错误。B不行，数组名不能直接等。C不行，*s只能赋值char。D可以。
6. `D`
   A正确，B正确，C正确, D错误（最短路径生成树不等于最小生成树）
7. `A`
   BFS+记录访问过的节点，直到所有节点都访问过。
8. `B`
   19 16 23 1 68 20 84 27 55 11 10 75
9. `D`
   判定树形状只跟元素个数有关.
10. `B`
    0 1 2 3 4 5
    r f
    r f
    r f
11. `B`
    767 - 511 = 256 对应128个父亲节点不是叶子了
    最后一层还剩下128个叶子
    256 + 128 = 384
    编程验证 x11.c
12. `A`
    深度
13. `D`
    Hash不用关键字比较
14. `B`
    快排不稳定
15. `A`

选择排序，每次选出后面n-i+1个里边最小的，放进第i个位置

填空题

1. `HDEBFGCA`
   编程验证。 t1.c
2. `6`

   57 11 12 -23 76 1901 20 96 -38 0
   = < < < > > < > < <
3. `17`
   一共需要合并8次，每合并1次，多一个节点
4. `最小生成树`
5. `BCE`
   [A]
   [AB]
   [A], B
   [AC]
   [A], C
   [AD]
   [ADE]
   [AD], E

编程题参考题解

题解非官方，个人向，由AC的同学提供。

1. 空 闲 空 间 合 并

本题为结构体排序题目的变种。在这里提供一种方法。首先对于所有块进行排序，然后从小到大开始遍历，判断是否能合并。能够合并则后面被合并的块记无效，前面的将end后延；最后只需要输出有效的块即可。在这里主要受排序的复杂度所限。下面程序给出的是冒泡排序进行的结构体排序，时间复杂度为O(n^2)

```
#include <stdio.h>

struct block {
    int able;
    int start;
    int end;
};

int main() {
    int num;
    scanf("%d", &num);
    struct block array[105];
    int i, j;
    for (i = 0; i < num; i++) {
        array[i].able = 1;
        scanf("%d%d", &array[i].start, &array[i].end);
    }
    for (i = 0; i < num; i++) {
        for (j = 0; j < num - i - 1; j++) {
            if (array[j].start > array[j + 1].start) {
                struct block temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
    int temp = array[0].end;
    int mark = 0;
    for (i = 1; i < num; i++) {
        if (temp + 1 == array[i].start) {
            temp = array[i].end;
            array[mark].end = temp;
            array[i].able = 0;
        } else {
            temp = array[i].end;
            mark = i;
        }
```

```
    }
    for (i = 0; i < num; i++) {
        if (array[i].able == 1) {
            printf("%d %d\n", array[i].start, array[i].end);
        }
    }
    return 0;
}
```

2. 火车货运调度模拟

在这里，这位同学采用了将常见数据结构操作进行封装的方式进行实现，这样使得最终代码较为简洁。

这道题是一道关于栈的模拟。按照题意进行模拟即可

stack.h

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAxSIZE 200

typedef struct stack{
  void *data[MAxSIZE];
  int log_len,alloc_len;
}stack;

void creat_stack(stack* a){
  a->log_len=0;
  a->alloc_len=MAxSIZE;
  return;
}

void stack_push(stack* a,void *adder){
  if(a->alloc_len==a->log_len){
    printf("stackfull\n");
    return;
  }
  a->data[a->log_len]=adder;
  a->log_len++;
  return;
}

void stack_pop(stack *s,void **cup){
  if(s->log_len<=0){
    return;
  }
  *cup=s->data[--s->log_len];
  return;
}

```

train.c

```
#include <stdio.h>
#include "stack.h"

```

```
#define MAXSIZE 100

typedef struct {
    char mane[20];
    int le;
}Map;
typedef struct{
    int le,x;
    char num[10]
} TRAIN_node;

Map map[MAXSIZE];
TRAIN_node nodes[MAXSIZE],result[MAXSIZE];

int cmp(const void*,const void *);

int main() {
    TRAIN_node *cup;
    stack Aline, Bline, Cline;
    creat_stack(&Aline);
    creat_stack(&Bline);
    creat_stack(&Cline);
    char s[20];
    int i, j;
    int n_place, n_node,count=0,progress=0;
    scanf("%d", &n_place);
    for (i = 0; i < n_place; ++i) {
        scanf("%s%d",  map[i].mane, &map[i].le);
    }
    scanf("%d", &n_node);
    for (i = 0; i < n_node; ++i) {
        scanf("%s%s", s, &nodes[i].num);
        nodes[i].x = i;
        for (j = 0; j < n_place; ++j) {
            if (strcmp(s, map[j].mane) == 0) {
                nodes[i].le = map[j].le;
                break;
            }
        }
    }
    memcpy(result,nodes,sizeof(TRAIN_node)*MAXSIZE);
    qsort(result,n_node,sizeof(TRAIN_node),cmp);
    for (i=n_node-1;i>=0;--i)
    {
        stack_push(&Bline,&nodes[i]);
        count++;
    }
    while (Bline.log_len!=0){
        do {
```

```
            stack_pop(&Bline,&cup);
            stack_push(&Aline,cup);
        }while (strcmp(cup->num,result[progress].num)!=0);
        stack_pop(&Aline,&cup);
        stack_push(&Cline,cup);
        while (Aline.log_len>progress){
            stack_pop(&Aline,&cup);
            stack_push(&Bline,cup);
            count++;
        }
        progress++;
        stack_pop(&Cline,&cup);
        stack_push(&Aline,cup);
    }
    for (i = 0; i < n_node; ++i) {
        printf("%s ",result[i].num);
    }
    printf("\n%d",count);
    return 0;
}

int cmp(const void* a,const void* b) {
    if (((TRAIN_node *) a)->le > ((TRAIN_node *) b)->le)
        return -1;
    else if (((TRAIN_node *) a)->le < ((TRAIN_node *) b)->le)
        return 1;
    else {
        return ((TRAIN_node *) a)->x - ((TRAIN_node *) b)->x;
    }
}
```

3. 查找同名文件

构建文件树

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXSIZE 100

typedef struct afile{
    int time,type;
    char mane[20];
    int flag;
    struct afile* link[MAXSIZE];
    struct afile* parent;
}afile;
typedef struct rtype{
    afile *result;
    int deep;
}result;

result results[MAXSIZE];
char Target[20];
afile *root,files[10000]={};
int count=0;

void dfs(afile*,int );
int cmp(const void *,const void *);
void print(afile*);

int main() {
    FILE *fpr=fopen("files.txt","r");
    int i,j;
    int n_f;
    char fname[20];
    scanf("%s%d",Target,&n_f);
    for (int i = 0; i < n_f; ++i) {

 fscanf(fpr,"%s%s%d%d",&files[i].mane,fname,&files[i].type,&files[i].time)
;
        if (strcmp(fname,"*")==0)
            root=&files[i];
        else{
            for (j = 0; j < i; ++j) {
                if (strcmp(fname,files[j].mane)==0){
                    files[i].parent=&files[j];
                    files[j].link[files[j].flag++]=&files[i];
```

```
                }
            }
        }
    }
    dfs(root,0);
    qsort(results,count,sizeof(result),cmp);
    for (i = 0; i < count; ++i) {
        print(results[i].result);
    }
    return 0;
}

void dfs(afile* a,int deep) {
    int i;
    if (a->type == 0) {
        if (strcmp(a->mane, Target) == 0) {
            results[count].result = a;
            results[count++].deep = deep;
        }
        return;
    } else {
        for (i = 0; i < a->flag; i++) {
            dfs(a->link[i], deep + 1);
        }
    }
}

int cmp(const void *a,const void *b) {
    if (((result *) a)->result->time > ((result *) b)->result->time)
        return 1;
    if (((result *) a)->result->time < ((result *) b)->result->time)
        return -1;
    else {
        return ((result *) a)->deep - ((result *) b)->deep;
    }
}

void print(afile *i){
    if (i->parent!=NULL){
        print(i->parent);
    } else{
        printf("%s:",i->mane);
        return;
    }
    printf("\\%s",i->mane);
    if (i->type==0)
        printf("\n");
    return;
}
```