考试说明：

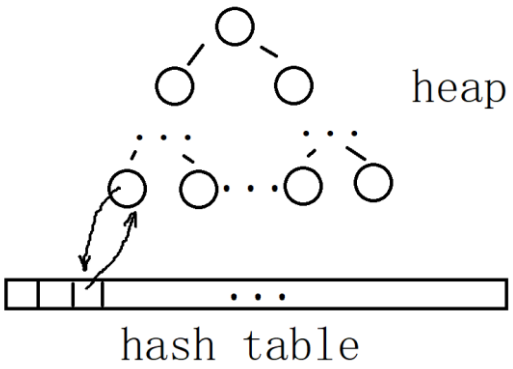
程序代码必须使用 C/C++语言完成，不接受任何其他编程语言。

以下有三道大题，第一题是必做题，剩下两题任选其一完成考试。

第一题总分 50 分，第二题总分 50 分，第三题总分 50 分。

不鼓励同时完成三道大题，否则在后两题中取较低分数统计总分，请特别注意。

一、利用以下数据结构解决定时队列问题。



假设某一服务器要处理若干定时任务。这些任务按照时间的先后离开优先权队列（用最小堆实现，如图所示），出队列的任务被立即执行。

定时任务的数据类型如：

```
struct TimedTask{
    long id;
    time_t t;
    ...
};
```

t 是最小堆中排序的关键字，即拥有最小 t 的任务位于堆顶。

同时，用户可以在任务被执行之前撤销任务，或者修改任务的执行时间。为了实现这样的操作，另外准备了一个散列表，用于计算散列地址的关键字是 id，散列表中每个元素是 id 和相应任务在最小堆中的下标的二元组（{id, idx}）。

具体要求如下：

(1) 由于任务是动态加入的，所以要实现将新任务加入最小堆的操作，即实现 bool adjustUp(TimedTask task_queue[],int pos,HashItem hash_table[], int hsize);各参数的意义比较明确，返回值为 true 则代表在上推的过程中发生了数据交换。（10 分）

(2) 堆顶任务的时间一到，该项任务就会出队列，此时要将剩下的任务重新组织成堆，因此要实现 bool adjustDown(TimedTask task_queue[],int pos,int qsize, HashItem hash_table[],int hsize);各参数的意义比较明确，返回值为 true 则代表在下推的过程中发生了数据交换。（10 分）

(3) 在堆中的任意操作（插入、删除、修改、调整），都有可能改变一些任务在堆中的位置，因此要实现散列表的插入、删除、修改等操作（线性开地址法）。提示：堆上的 adjustUp() 和 adjustDown()函数都要考虑散列表上的操作。（8 分）

(4) 在任意时刻，可以删除堆中任意位置上的任务（通过 id 定位）。此时，要以最小的代价迅速将剩下的任务重新组织成最小堆，并完成散列表上相应的修改。即完成：void deleteFromHeap(TimedTask task_queue[],int qsize,HashItem hash_table[],int hsize,long id)；（8 分）

(5) 在任意时刻，可以修改堆中任意位置上的任务（通过 id 定位）的时间 t，此时，要

从被修改的任务开始迅速将所有任务重新组织成一个最小堆，并完成散列表上相应的修改。即完成：void updateHeap(TimedTask task_queue[],int qsize,HashItem hash_table[],int hsize,long id,time_t t)；（8 分）

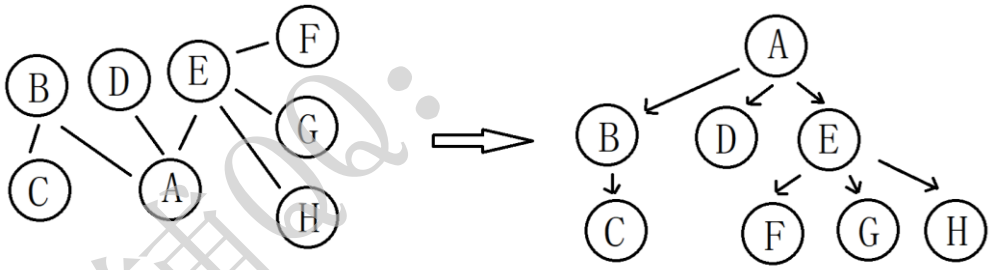
(6) 依次有如下任务进队列：(1237585, 7875) (237585, 7855) (127585, 875) (17585, 975) (47585, 17875) (585, 75)。假设最后一个任务进队列时并没有一个任务被执行，请输出优先权队列以及散列表的内容（假设散列表的长度为 7）。（6 分）

要求 main()函数实现下述操作：

```
int main(){
    initializeHashTable(hashtable,7);
    buildQueue(taskqueue,6,hashtable,7);
    dispQueue(taskqueue,6);
    updateHeap(taskqueue,6,hashtable,7,237585,700);
    dispQueue(taskqueue,6);
    deleteFromHeap(taskqueue,6,hashtable,7,17585);
    dispQueue(taskqueue,5);
    return 0;
}
```

要求补充实现所有可能需要用到的函数和数据结构。

二、输入一个无向图的邻接矩阵，判断该无向图是否为一个无根无向树。若是，则设计一个算法，将这棵无根无向树转换成一个具有最小高度的有根有向树，并输出其后序序列。



具体要求如下：

(1) 将以上左边部分的图表示成邻接矩阵的形式，并作为算法的输入数据。（5 分）

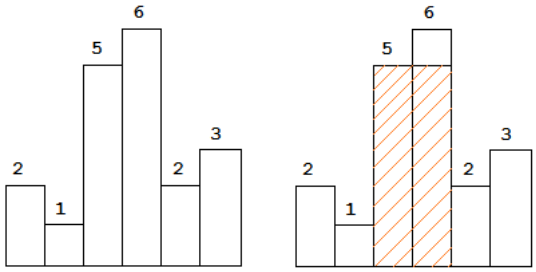
(2) 设计一个算法，能判断一个邻接矩阵是否为一棵无根无向树。（10 分）

(3) 若是无根无向树，则设计一个算法，在指定某顶点为树根的情况下，将邻接矩阵转换成一棵用孩子兄弟链存储的有根有向树。（20 分）

(4) 输出有根有向树的后序遍历序列。（5 分）

(5) 设计一个算法，寻找最优的树根（导致高度最小的有根有向树）。在找到最优根之后，生成拥有最小高度的有根有向树并输出其后序序列。提示：分三步进行。第一步，求所有顶点对之间的路径长度（参考 Floyd 算法），求出最大的路径长度；第二步，根据这个最大路径长度，深度优先遍历找到该路径上的所有顶点；第三步，在该路径上选择最优根。（10 分）

三、给定 *n* 个非负整数，用来表示柱状图中各个柱子的高度，每个柱子彼此相邻，且宽度为 1。求在该柱状图中，能够勾勒出来的矩形的最大面积。



例如，上图中阴影部分是能够勾勒出来的最大矩形，面积为 10。

具体要求如下：

(1) 函数原型为 int maxRectangle(int A[],int n)，A 是存储柱状图的数组，n 为数组长度。

(2) 用非递归方式实现该算法。

(3) 在算法运行的过程中，允许改变数组 A 的内容。

(4) 提示：数组为递增时（其实递减的情况类似），最容易求得问题的解，每一根柱子都可以向右扩展。因此，可以先尝试在柱状图的高度递增的情况下的问题的解。（30 分）

(5) 然后，可用一个递增栈（即栈中的元素按增序排列）作为辅助数据结构，解决在任意形状的柱状图上求最大矩形面积的问题。（20 分）