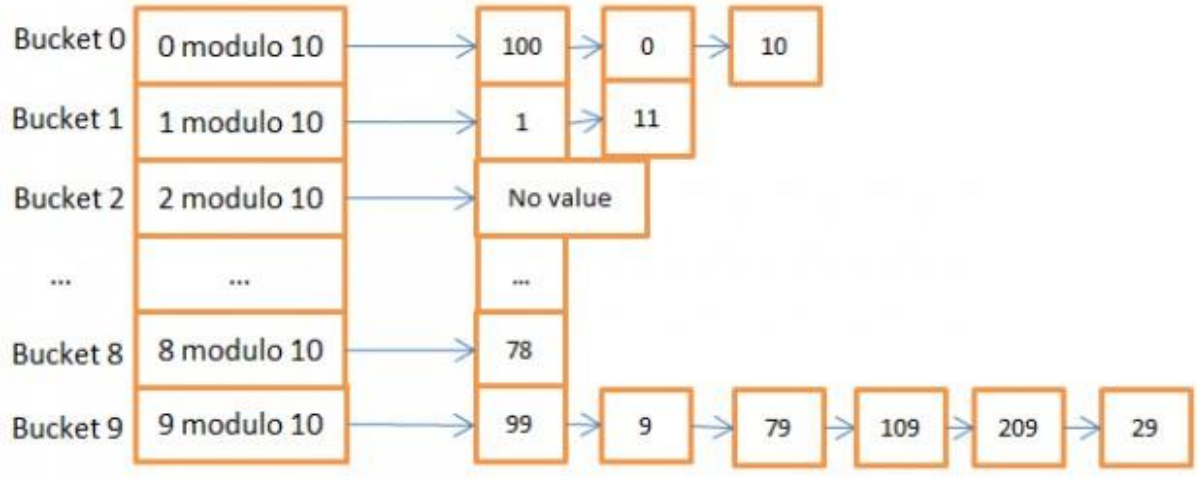
Lab 0：哈希表与跳表

一、 实验介绍

哈希表(Hash Table)，又叫散列表，是一种提供了快速查找和插入操作的数据结构。哈希表基于数组下标定位的思想，实现了某种将键转换为数组下标的函数 f(key) ，叫做**哈希函数**(Hash Function)，例如，在英文字典中查询一个单词，我们可以按照字母序迅速定位到单词在字典中的范围，这就是哈希函数的一种实际应用。

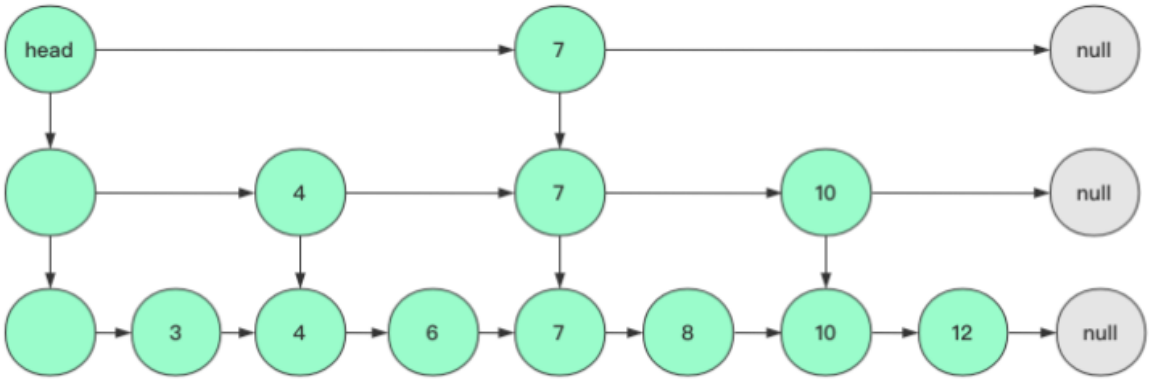
哈希函数将给定键值映射到哈希表范围中的某个位置，如果两个键值被映射到同一位置，称其产生了一个哈希冲突。本实验中，我们采用**链地址法**或者叫做**哈希桶法**来解决哈希冲突。它的基本思想是，将具有相同哈希值的数据连接到一个单向链表中，哈希表对应哈希值下标的位置，我们称作哈希桶(Bucket)，这个桶中存放链表的头结点。

如下图所示，是一个使用取余哈希函数的哈希表结构。



跳表(Skiplist)是有序元素序列快速搜索查找的一个数据结构。它利用了**二分查找**的思想，对于一个**有序**的链表，可以先与中间节点比较从而迅速筛去一半的节点，如此往复在对数时间内完成查找工作。于是我们可以为部分链表节点添加上层索引，通过元素值比较来跳过额外的底层节点，这些索引又可以再创建更上层的索引来跳过更多节点，通过这种多层索引方式实现的数据结构就是跳表。

如下图所示，是一种跳表实现的数据结构的示意图。



二、 实验要求

本次 Lab 中，我们要求使用 C++实现哈希表和跳表这两种数据结构。

1. 哈希表

在 HashTable.h 中，我们已经定义好了节点类型 HashNode 和哈希表类 HashTable。HashNode 为我们要操作的数据节点，它包括一个 <key, value> 键值对以及指向下一节点的指针。HashTable 中定义了对应的哈希桶数组，还定义了哈希表的一些操作函数，实验要求实现主要操作函数的部分，它们的实现在 HashTable.cpp 中，你需要完成的部分如下：

① **void HashTable::Insert (int key, int value)**：将键为 key 值为 value 的节点插入到哈希表中，如果已经存在 key，则更新它的值。为了保证大家的实现结果一致，请使用我们提供的统一哈希函数来决定将节点插入到哪个哈希桶中。新插入节点应该位于对应桶中链表的结尾。

② **void HashTable::Search (int key)**：在哈希表中查找键为 key 的节点，输出它的桶编号、该节点在桶中链表的序号(从 0 计)、该节点的 key 和 value 值。如果没有找到，输出“Not Found”。

③ **void HashTable::Delete (int key)**：将键为 key 的节点从哈希表中删去，如果节点不存在，则直接返回。

2. 跳表

跳表部分的实现，请同学们阅读课程网站上的 skiplists 论文学习算法思路，本实验将基于此来实现。同样地，在 SkipList.h 中，我们定义好了 SKNode 节点和跳表类 SkipList。SKNode 中包括节点类型(HEAD, NORMAL 或 NIL)、键值对以及 forward 指针数组，该数组存储每层 list 中该节点的后继节点指针，大小为宏定义 MaxLevel，后续实现也请使用该宏定义。SkipList 中定义了头尾节点及主要操作函数，类似地你需要在 SkipList.cpp 中完成以下部分的实现：

① **void SkipList::Insert (int key, int value)**：将键为 key 值为 value 的节点插入到跳表中，如果已经存在 key，则更新它的值。由于新节点的插入涉及到随机化，为了使大家实现结果一致，同样请使用我们提供的统一伪随机函数。

② **void SkipList::Search (int key)**：在跳表中查找键为 key 的节点，输出查找过程途径所有节点的层数和 key 值，如果是 head 节点或 NIL 节点， key 值用 'h' 或 'N' 代替。如果找到，再输出 value 。如果没有找到，再输出“Not Found”。

③ **void SkipList::Delete (int key)**：将键为 key 的节点从跳表中删去，如果节点不存在，则直接返回。

3. 输入输出

(1) 输入部分

上述数据结构初始化都为空，本次实验的输入部分通过文件给出，它是一个对应操作的指令序列，每行一个指令依序执行，如下图所示：

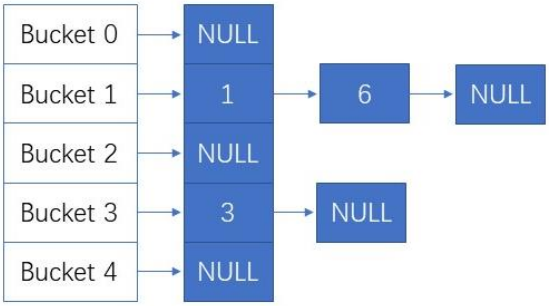
```
Insert(1,123)
Insert(3,456)
Insert(6,789)
Search(3)
Search(5)
Search(6)
```

(2) 输出部分

如果按照本节前面实验要求提到的输出，你的输出应具有以下格式。

哈希表部分：

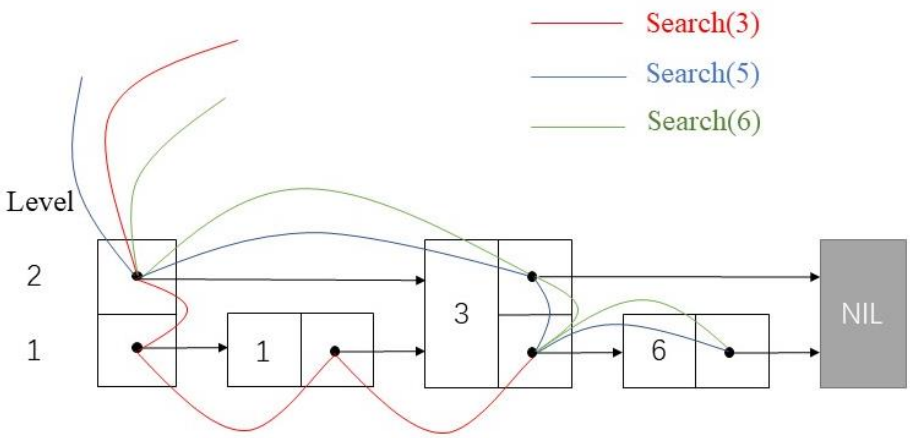
执行 Search 函数时输出桶编号，节点序号，key 和 value 值，均以空格隔开，每行末尾不加空格。Display() 的输出，这个无需同学们添加，我们会在哈希表部分输入执行完成后自动调用来检查哈希表结构是否正确。以上述指令序列为例，假设桶数为 5，对应哈希表和输出如下(仅作示例，不代表实际结果)：




```
bucket 3 index 0 key 3 value 456
Not Found
bucket 1 index 1 key 6 value 789
|Bucket 0|-->
|Bucket 1|-->(1,123)-->(6,789)-->
|Bucket 2|-->
|Bucket 3|-->(3,456)-->
|Bucket 4|-->
```

跳表部分：

执行 Search 函数时在一行输出途径节点的层数和 key 值，如果是 head 节点或 NIL 节点， key 值处输出 “h” 或 “N” ，层数和 key 之间用 “,” (英文半角逗号)隔开，每组之间用空格隔开，每行末尾不加空格。Display() 的输出，由我们自动添加。以上述指令序列为例，对应跳表和输出如下(仅作示例，不代表实际结果)：




```
2,h 1,h 1,1 1,3 456
2,h 2,3 1,3 1,6 Not Found
2,h 2,3 1,3 1,6 789
Level 2:h-->(3,456)-->N
Level 1:h-->(1,123)-->(3,456)-->(6,789)-->N
```

三、 实验评分

1. 代码运行

我们提供了 Makefile 工具，在主目录下执行 **make** 指令生成可执行程序 main。你可以通过下述指令查看自己的实现情况：

```
./main h(或 s) 输入文件的路径
```

h 指哈希表，s 指跳表。

例如

```
./main s ./input/skiplist_input1
```

将使用 input 文件夹下的 skiplist_input1 文件中的指令序列用你的实现生成一个跳表，并将输出打印到标准输出中。你可以据此定义自己的指令序列用于 debug。

2. 代码评分

你可以在主目录下执行 **make grade** 指令查看自己实现是否正确，如果不正确，会打印

出你的输出与答案的差异，请检查是否正确实现以及按要求输出了，如果正确实现，你应该看到如下输出：

```
<<<<<<<< grade test >>>>>>>>
  Hash_1:    OK
  Hash_2:    OK
  Skip_1:    OK
  Skip_2:    OK
  grade: 100/100
<<<<<<<< grade test over >>>>>>>>
```

我们提供了哈希表和跳表的各 2 组数据集，通过它们你将拿到该实验内容的部分分数。实验评分会有额外的数据集，正确通过将得到其余分数，所以**请勿针对测试数据编程**。

3. 性能比较书面报告

在上述实验过程中，我们可以看出哈希表和跳表都可以实现对于键值数据的插入、查询和删除功能。在实际使用中，我们经常需要在这两类数据结构中进行选择。**假设我们在为某个应用挑选数据结构，该应用除了单个键值数据的插入、查询和删除外，还需要进行一段连续区间内的键值查询**。例如，列出所有 key 范围在 100 到 500 之间的键值对。在这种情况下，我们应该选择哈希表还是跳表？

请在书面报告中给出你的选择和原因分析（一两句话简述即可，100 字以内），然后通过实验的方法，给出实验数据证明你的分析（包括数据和简单的结论）。你可能需要按照我们上述输入文件的格式自制一份可以支撑你的结论的测试用例，请将实验数据绘制成一份图表，该图表要求能直观反映出两种数据结构在不同长度的区间查询中的性能（查询时间）变化。

请将上述内容生成一个 PDF 文件。

你可能需要修改代码添加输出来收集实验数据，由于我们会使用输出判断实现的正确性，请注意保存一份能够通过先前 grade test 的版本。

4. 实验上交

原则上，除了书面报告部分，你不应该修改除了 HashTable.cpp 和 SkipList.cpp 之外的代码。**提交实验请将 HashTable.cpp 和 SkipList.cpp 打包上传 Canvas，命名使用“学号+姓名+lab0”，如“520123456789+张三+lab0.zip”**。

性能比较书面报告同样上传到 Canvas 平台（会有单独的作业用于提交），要求 PDF 格式，命名无要求。

代码请提交能够通过 grade test 的版本，不要求提交性能比较的代码。

Lab0 截止日期为 2022.03.08 21:59PM。

四、 注意事项

1. 切勿**抄袭**！如若发现明显雷同现象，该实验评分将以 0 分计。
2. 注意实验截止日期，迟交或不交会视情况扣分。
3. 请注意按照实验要求完成，避免因为自动脚本评分造成的判分失误。
4. 有问题随时通过邮件或微信群联系负责 Lab 的助教姬浩迪、陈沛东。