**第四次作业**

1. **程序实现**
2. 实现基于动态数组和链表的字符串，并添加如下功能：
3. 对于链表字符串，增加字符串重构函数：void linkString::reconstruct(int nodeSize)，用于实现多次插入和删除后的结点容量调整（可以为，也可以为其它整数）；
4. 对于数组字符串和链表字符串，增加字符频率统计函数：void frequency()，输出字符串中各个不同字符出现的频率。
5. 对于数组字符串和链表字符串，增加基于KMP算法（P151）的字符串查找函数：int find(const string &sub)const，如找到，返回第一次出现的位置，否则返回-1；(可以先自习，等下次课讲完再做)
6. 对于数组字符串和链表字符串，增加基于查找函数的字符串替换函数：void replace(const string &s1, const string &s2)，将当前字符串中所有与s1相同的子串替换成s2。
7. **性能测试**
8. 对基于链表实现的字符串，测试其性能：
9. 创建一个linkString对象，执行50000次插入操作，每次插入20个任意字符，使其成为一个较长的字符串；
10. 设置结点的容量为不同整数，使用reconstruct()函数重构块状链表，分别执行1000次的随机插入、删除和查找操作，测量插入时间、删除时间和查找时间（注意删除时避免将字符串删空）；验证现有的研究结果，即块状链表的结点容量与结点个数相同时算法的效率是最高的。
11. 对基于数组实现的字符串，测试其性能：
12. 创建一个seqString对象，执行50000次插入操作，每次插入20个任意字符，使其成为一个较长的字符串；
13. 分别执行1000次的随机插入、删除和查找操作，测量插入时间、删除时间和查找时间（注意删除时避免将字符串删空），与块状链表比较性能差异。