20190408-左士海-作业 3

1. 给出一个算法，求循环链表中结点的个数。

```
79  template<class T>
80  int LinkList<T>::listLength() {
81      int num = 0;
82      Node<T> * p = head->next;
83      while (p != head) {
84          p = p->next;
85          num++;
86      }
87      return num;
88  }
```

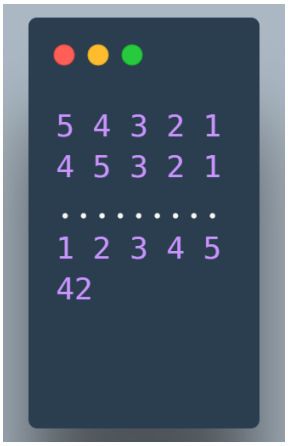
循环链表对于单链表的区别在于：尾指针 next 域指向头指针。所以遍历完成的标志由尾指针 next 域指向 nullptr，变为指向头指针。
时间复杂度：O(n)。

2. 试编写算法，将元素为整数的顺序表 $(a_1, a_2, \ldots, a_n)$ 重新排列为以 a_1 为界的两部分：a_1 前面的值均比 a_1 小，a_1 后面的值都比 a_1 大，假定各个元素值互不相等。要求时间复杂度为 $O(n)$。

```
36      void splitByA(int index) { splitByA(data, index - 1); }
37      void splitByA(T data[], int index) {
38          T temp = data[index];
39          int pos = 0;
40          T t;
41          for (int i = 1; i < length; ++i ) {
42              if (data[i] < temp) {
43                  pos++;
44                  t = data[i]; data[i] = data[pos]; data[pos] = t;
45              }
46          }
47          if (pos > 0) {
48              t = data[index];
49              data[index] = data[pos];
50              data[pos] = t;
51          }
52      }
53  };
```

从左向右遍历，用 pos 记录不大于a_1的最后一位的索引，则 pos+1 位大于a_1的第一个，若当前值小于a_1，与 pos+1 的进行交换。最后 pos 为a_1的位置。

3. 编号为 1、2、3、4、5 的五辆列车，顺序开进一个栈式结构的站点，问开出车站的顺序有多少种可能？请具体写出所有可能的出栈序列。



深度优先搜索，在任何时间，条件允许时都可以进行出栈或是入栈操作。栈不为空可以出栈，序列不为空可以进栈。答案是共有 42 种。

```
// 入栈的个数 i, 出栈个数 j, 序列总数 n, 栈 stk, 当前出栈序列 now, 总个数 ans
void dfs(int i, int j, int n, int input[], LinkStack<int> & stk, int now[], int & ans)
{   if(i == n) {     // 全部入栈
        if(!stk.Empty()) {  // 栈不为空 只能出栈
            int x = stk.Pop();
            now[j + 1] = x;
            dfs(i, j+1, n, input, stk, now, ans);
            stk.Push(x);
            now[j + 1] = 0;
        } else {
            ans++;
            for (int k = 1; k <= i; ++k) {
                cout << now[k] << " ";
            }
            cout << endl;
        }
    } else {         // 可以选择出栈或者入栈
        // 入栈
        stk.Push(input[i + 1]);
        dfs(i + 1, j, n, input, stk, now, ans);
        stk.Pop();
        // 出栈
        if(!stk.Empty()) {
            int x = stk.Pop();
            now[j + 1] = x;
            dfs(i, j + 1, n, input, stk, now, ans);
            stk.Push(x);
            now[j + 1] = 0;
        }
    }
}
```

4. 利用栈实现把十进制整数转换为二至十六之间的任一进制数输出的功能。

```
const char h[] = {'A', 'B', 'C', 'D', 'E', 'F'};
// 存储结果栈：stk, 原十进制：val 要转化的进制：n
void tenToN(LinkStack<int> stk, int val, int n) {
    if(val == 0) {
        while (!stk.Empty()) {
            int x = stk.Pop();
            if(x < 10) {
                cout << x;
            } else {
                cout << h[x%10];
            }
        }
        cout << endl;
        return;
    }
    stk.Push(val % n );
    val /= n;
    tenToN(stk, val, n);
}
```

十进制转其他进制实际是不断取余的过程再倒序输出，与栈先进后出的性质相吻合，通过递归，将每次的余数压入栈，最后依次出栈。递归的过程时间复杂度根号级，加输出环节，总时间复杂度 O(n)。

5. 设有一维数组 stack[StackMaxSize]，分配给两个栈 S1 和 S2 使用，如何分配数组空间，使得对任何一个栈，当且仅当数组空间全满时才不能插入。试说明你的分配方法。并分别给出两个栈各自的入栈和出栈算法。

```cpp
template<typename T, int MaxSize>
class Stack {
    T data[MaxSize];
    int top1, top2;
public:
    Stack() {
        top1 = -1;
        top2 = MaxSize;
    }
    void pushToStack1(const T& item) {
        if (top1 + 1 == top2) {
            cerr << "上溢";
            exit(1);
        }
        data[++top1] = item;
    }
    T popFromStack1() {
        if (top1 == -1) {
            cerr << "下溢";
            exit(1);
        }
        return data[top1--];
    }
    T Top1() {
        if(top1 == -1) {
            cerr << "下溢" << endl;
            exit(-1);
        }
        return data[top1];
    }
    void pushToStack2(const T & item) {
        if (top2 - 1 == top1) {
            cerr << "下溢";
            exit(-1);
        }
        data[--top2] = item;
    }
    T popFromSatck2() {
        if (top2 == MaxSize) {
            cerr << "下溢";
            exit(-1);
        }
        return data[top2++];
    }
    T Top2() {
        if (top2 == MaxSize) {
            cerr << "下溢";
            exit(-1);
        }
        return data[top2];
    }
};
```

共享存储空间栈，定义两个 top 索引，分别指向数组的首尾，栈一入栈时 top1++，出栈时 top1--，栈二入栈时 top2--，出栈时 top2++。若 top1+1 = top2，或者 top2-1 = top1，则表示栈满。

6. 假设表达式中允许包含 3 种括号：圆括号、方括号和大括号。试编写一个算法，检查表达式中括号是否配对，若能够全部配对则返回 1，否则返回 0。

```
int checkBracket(string str) {
    LinkStack<char> stk;
    int len = str.length();
    int i = 0;
    while (i < len) {
        switch (str[i]) {
            case '[': case '(': case '{':
                stk.Push(str[i++]);
                break;
            case ']': case ')': case '}':
                if (!stk.Empty()) {
                    char e = stk.Pop();
                    if ((str[i] == ']' && e != '[')
                            || (str[i] == ')' && e != '(')
                            || (str[i] == '{' && e != '}')) {
                        return 0;
                    } else {
                        i++;
                        break;
                    }
                } else {
                    return 0;
                }
            default:
                i++;
        }
    }
    return 1;
}
```

依次读入括号字符，其满足先遇到某括号，后遇到与其匹配的反括号，这特性与栈先进后出，后进先出的性质相吻合，栈用于储存左括号，栈顶一定是最后的左括号，必先遇到与其匹配的右括号，进行判断即可。

总时间复杂度：O(n)。