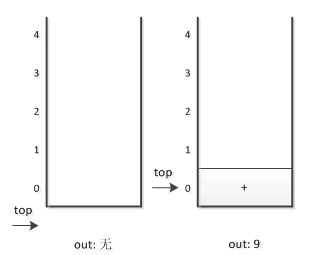
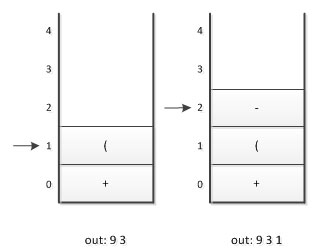
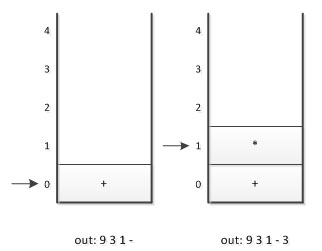
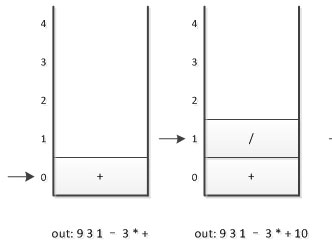
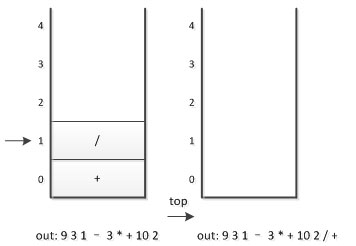
**[将中缀表达式转化为后缀表达式](http://www.nowamagic.net/librarys/veda/detail/2307)**

* 规则：从左到右遍历中缀表达式的每个数字和符号，若是数字就输出，即成为后缀表达式的一部分；若是符号，则判断其与栈顶符号的优先级，是右括号或优先级低于找顶符号（乘除优先加减）则栈顶元素依次出找并输出，并将当前符号进栈，一直到最终输出后缀表达式为止。
* [中缀表达式](http://www.nowamagic.net/librarys/veda/tag/中缀表达式)“9+(3-1)\*3+10/2”转化为后缀表达式“9 3 1-3\*+ 10 2/+”
* 下面我们来具体看看这个过程。
* 1. 初始化一空栈，用来对符号进出栈使用。
* 2. 第一个字符是数字9，输出9，后面是符号“+”，进[栈](http://www.nowamagic.net/librarys/veda/tag/栈" \t "_blank)。
* 
* 3. 第三个字符是“(”,依然是符号，因其只是左括号，还未配对，故进栈。
* 4. 第四个字符是数字3，输出，总表达式为9 3,接着是“-”进栈。
* 
* 5. 接下来是数字1，输出，总表达式为9 3 1，后面是符号“)”，此时，我们需要去匹配此前的“(”，所以栈顶依次出栈，并输出，直到“(”出栈为止。此时左括号上方只有“-”，因此输出“-”，总的输出表达式为9 3 1 -
* 6. 接着是数字3，输出，总的表达式为9 3 1 - 3 。紧接着是符号“\*”，因为此时的栈顶符号为“+”号，优先级低于“\*”，因此不输出，进栈。
* 
* 7. 之后是符号“+”，此时当前栈顶元素比这个“+”的优先级高，因此栈中元素出栈并输出（没有比“+”号更低的优先级，所以全部出栈），总输出表达式为 9 3 1 - 3 \* +.然后将当前这个符号“+”进栈。也就是说，前6张图的栈底的“+”是指中缀表达式中开头的9后面那个“+”，而下图中的栈底（也是栈顶）的“+”是指“9+(3-1)\*3+”中的最后一个“+”。
* 8. 紧接着数字10，输出，总表达式变为9 3 1-3 \* + 10。
* 
* 9. 最后一个数字2,输出，总的表达式为 9 3 1-3\*+ 10 2
* 10. 因已经到最后，所以将栈中符号全部出栈并输出。最终输出的后缀表达式结果为 9 3 1-3\*+ 10 2/+
* 

|  |  |  |
| --- | --- | --- |
| 01 | // <Copyright liaoqb> | |
| 02 | #include <iostream> |

|  |  |
| --- | --- |
| 03 | #include <stack> |
| 04 | #include <string> | |

|  |  |  |
| --- | --- | --- |
| 05 | #include <queue> | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | using namespace std; | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | int main(int argc, char\* argv[]) { | |
| 10 | int t; |

|  |  |
| --- | --- |
| 11 |  |
| 12 | cin >> t; | |

|  |  |
| --- | --- |
| 13 |  |
| 14 | while (t--) { | |

|  |  |
| --- | --- |
| 15 | string str; |
| 16 | queue<char> q; | |

|  |  |  |
| --- | --- | --- |
| 17 | stack<char> s;  // operation stack | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | cin >> str; | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | for (unsigned i = 0; i < str.size(); ++i) { |
| 22 | if ((str[i] >= 'A' && str[i] <= 'Z') || (str[i] >= 'a' && str[i] <= 'z')) { | |

|  |  |  |
| --- | --- | --- |
| 23 | q.push(str[i]);  // if it is letters, push in the queue | |
| 24 | } else if (str[i] == '(') { |

|  |  |
| --- | --- |
| 25 | s.push(str[i]); |
| 26 | } else if (str[i] == ')') { | |

|  |  |
| --- | --- |
| 27 | while (s.top() != '(') { |
| 28 | q.push(s.top());  // dealing the bracket | |

|  |  |  |
| --- | --- | --- |
| 29 | s.pop(); | |
| 30 | } |

|  |  |
| --- | --- |
| 31 | s.pop(); |
| 32 | } else if (str[i] == '+' || str[i] == '-') { | |

|  |  |
| --- | --- |
| 33 | if (s.empty()) { |
| 34 | s.push(str[i]); | |

|  |  |  |
| --- | --- | --- |
| 35 | } else if (s.top() == '+' || s.top() == '-' || s.top() == '\*' || s.top() == '/') { | |
| 36 | do { |

|  |  |  |
| --- | --- | --- |
| 37 | q.push(s.top()); | |
| 38 | s.pop(); |

|  |  |  |
| --- | --- | --- |
| 39 | } while (!s.empty() && (s.top() == '+' || s.top() == '-' || s.top() == '\*' || s.top() == '/')); | |
| 40 | // pop to the queue |

|  |  |  |
| --- | --- | --- |
| 41 | s.push(str[i]); | |
| 42 | } else { |

|  |  |  |
| --- | --- | --- |
| 43 | s.push(str[i]); | |
| 44 | } |

|  |  |  |
| --- | --- | --- |
| 45 | } else if (str[i] == '\*' || str[i] == '/') { | |
| 46 | if (s.empty()) { |

|  |  |
| --- | --- |
| 47 | s.push(str[i]); |
| 48 | } else if (s.top() == '\*' || s.top() == '/') { | |

|  |  |
| --- | --- |
| 49 | do { |
| 50 | q.push(s.top()); | |

|  |  |
| --- | --- |
| 51 | s.pop(); |
| 52 | } while (!s.empty() && (s.top() == '\*' || s.top() == '/')); | |

|  |  |  |
| --- | --- | --- |
| 53 | s.push(str[i]); | |
| 54 | } else { |

|  |  |  |
| --- | --- | --- |
| 55 | s.push(str[i]); | |
| 56 | } |

|  |  |  |
| --- | --- | --- |
| 57 | } | |
| 58 | } |

|  |  |
| --- | --- |
| 59 |  |
| 60 | while (!s.empty()) { | |

|  |  |  |
| --- | --- | --- |
| 61 | q.push(s.top()); | |
| 62 | s.pop(); |

|  |  |  |
| --- | --- | --- |
| 63 | }  // if not empty, push to the queue | |
| 64 |  |

|  |  |
| --- | --- |
| 65 | while (!q.empty()) { |
| 66 | cout << q.front(); |

|  |  |
| --- | --- |
| 67 | q.pop(); |
| 68 | }  // cout it | |

|  |  |  |
| --- | --- | --- |
| 69 | cout << endl; | |
| 70 | } |

|  |  |
| --- | --- |
| 71 |  |
| 72 | return 0; | |

|  |  |
| --- | --- |
| 73 | } |