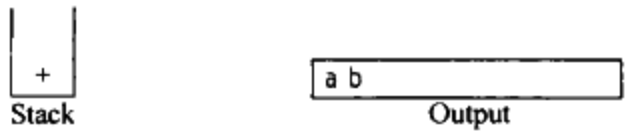
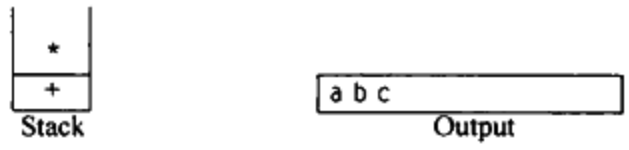
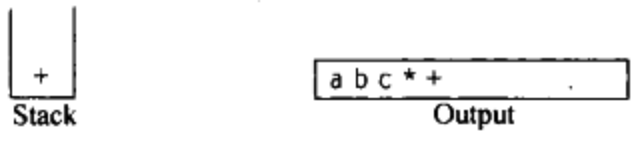


接着 * 号被读入。操作符栈的栈顶元素比 * 的优先级低，故没有输出且 * 进栈。接着，c 被读入并输出。至此，我们有

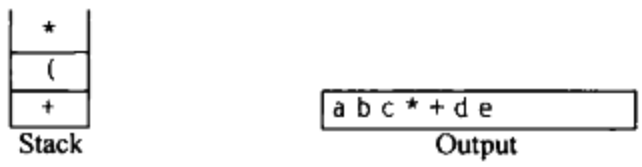


后面的符号是一个 + 号。检查一下栈我们发现，需要将 * 从栈弹出并把它放到输出中；弹出栈中剩下的 + 号，该算符不比刚刚遇到的 + 号优先级低而是有相同的优先级；然后，将刚刚遇到的 + 号压入栈中



下一个被读到的符号是一个(，由于有最高的优先级，因此它被放进栈中。然后，d 读入并输出

继续进行，我们又读到一个 * 。由于除非正在处理闭括号否则开括号不会从栈中弹出，因此没有输出。下一个是 e，它被读入并输出

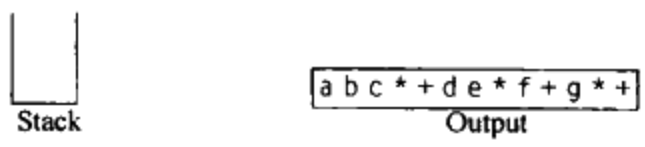


再往后读到的符号是 + 。我们将 * 弹出并输出，然后将 + 压入栈中。这以后，我们读到 f 并输出

现在，我们读到一个)，因此将栈元素直到(弹出，我们将一个 + 号输出

下面又读到一个 * ；该算符被压入栈中。然后，g 被读入并输出

现在输入为空，因此我们将栈中的符号全部弹出并输出，直到栈变成空栈



与前面相同，这种转换只需要 $O(N)$时间并经过一趟输入后工作完成。可以通过指定减法