2. First 12 items popped off stack:

1. (5,3)
2. (6,3)
3. (4,3)
4. (4,2)
5. (4,1)
6. (3,1)
7. (2,1)
8. (1,1)
9. (1,2)
10. (3,3)
11. (5,4)
12. (5,5)

3. First 12 items popped off the queue

1. (5,3)
2. (5,4)
3. (4,3)
4. (6,3)
5. (5,5)
6. (3,3)
7. (4,2)
8. (5,6)
9. (4,5)
10. (4,1)
11. (5,7)
12. (3,5)

In stacks, you visit each row with depth search first, following the path with the most recent data points on top. Once a dead end is hit, then you resume your previous path. With a queue, rather, you explore the oldest data point that has been in the list. Because of this, the order of which coordinates are popped off differ.

stack<**char**> operands;

**char** operand1;

**char** operand2;

**for** (**int** i = 0; i < postfix.size(); i++)

{

**char** ch = postfix.at(i);

**if** (isalpha(ch) || ch == '!')

operands.push(ch);

**else**

{

operand2 = operands.top();

operands.pop();

operand1 = operands.top();

operands.pop();

**if** (ch == '|')

{

**if** (trueValues.contains(operand2))

operands.push(operand2);

**else** **if** (trueValues.contains(operand1))

operands.push(operand1);

**else** // both are false so push whatever

operands.push(operand2);

}

**if** (ch == '&')

{

**if** (falseValues.contains(operand2))

operands.push(operand2);

**else** **if** (falseValues.contains(operand1))

operands.push(operand1);

**else** // they must be true so doesn't matter which operand you push on

operands.push(operand2);

}

}

}

**if** (trueValues.contains(operands.top()))

result = **true**;

**else**

result = **false**;