CS 32 Worksheet Week 4

Concepts: Stacks, Queues

1. (5 mins) Given a string of '(', ')', '[', and ']', write a function `isValid` to check if the input string is *valid*. Validity is determined by each '(' having a corresponding ')', and each '[' having a corresponding ']', with parentheses being properly nested and brackets being properly nested.

Examples:

```
isValid("[()([])[[([][])]]]") // true
isValid("((([(])))")  // false, since not properly nested
isValid("(()))") // false, since no corresponding '(' for last ')'
isValid("()[]") // true
```

```
bool isValid(string symbols) {















}
```

2. (5 mins) Write a function `reverseQueue` that reverses a queue Q in place using a reference. Only the following standard operations are allowed on queue:

1) `Q.push(x)` : Add an item x to the back of the queue.
2) `Q.pop()` : Remove an item from the front of the queue.
3) `Q.front()` : Return the item at the front of the queue
4) `Q.empty()` : Check if the queue is empty or not.

You may use an additional data structure if you wish.

Example:

```
queue<int> q({10, 20, 30, 40, 50, 60, 70, 80, 90, 100});
reverseQueue(q)
// q should now be {100, 90, 80, 70, 60, 50, 40, 30, 20, 10}
```

```
void reverseQueue(queue<int>& Q) {

}
```

3. (5 mins) Evaluate the following postfix expression and show your work:

```
9 5 * 8 - 6 7 * 5 3 - / *
```

4. (15 mins) Write a function `findNextInts` that takes in two integer arrays of size *n: sequence* and *results*. This function assumes that *sequence* already contains a sequence of positive integers. For each position *i* (from 0 to *n*-1) of *sequence*, this function should find the **smallest index *j* such that *j* > *i* and *sequence[j]* > *sequence[i]*, and put *sequence[j]* in *results[i]***; if there is no such *j*, put -1 in *sequence[i]*. Try to do this without nested for loops both iterating over the array! (Hint: `#include <stack>`). In other words, we want to store the nearest value appearing later in the array than the current one that is greater than it in the result.

Example:

```
int seq[] = {2, 6, 3, 1, 9, 4, 7 }; // Only positive integers!
int res[7];
findNextInts(seq, res, 7);
for (int i = 0; i < 7; i++) { // Should print: 6 9 9 9 -1 7 -1
  cout << res[i] << " ";
}
cout << endl;
```

Notice that the last value in *results* will always be set to -1 since there are no integers in *sequence* after the last one!

```
void findNextInts(const int sequence[], int results[], int n) {

}
```

5. (10 mins) Implement a Stack class using only queues as data structures. This class should implement the *empty*, *size*, *top*, *push*, and *pop* member functions, as specified by the standard library's implementation of stack. (The implementation will not be very efficient.)

```
class Stack {










}
```

6. (16 mins) Implement a Queue class using only stacks as data structures. This class should implement the *empty*, *size*, *front*, *back*, *push*, and *pop* member functions, as specified by the standard library's implementation of queue. (The implementation will not be very efficient.)

```
class Queue {
}
```