Question 1: You are reading numbers from standard input, and you are guaranteed that each number that you read is either bigger than all of the previous numbers or smaller than all of the previous numbers. Like [5, 6, 7, 3, 9, 2, 10, 0].

Suppose you want to print the numbers sorted in ascending order. What is the best data structure from the STL that you should use for reading and sorting the numbers?

Question 2: What is the big-O running time for the answer in Question 1 if there are *n* numbers?

Question 3: What is the output of the following program:

```
#include <cstdio>
using namespace std;

int main()
{
  int a;

  a = 263;
  printf("0x%x\n", a);
  return 0;
}
```

Question 4: What is the output of the following program:

```
#include <cstdio>
using namespace std;

int main()
{
  int a, b;

  a = 263;
  b = 3;
  printf("0x%x\n", a & b);
  return 0;
}
```

Question 5: What is the output of the following program:

```
#include <cstdio>
using namespace std;

int main()
{
  int a, b;

  a = 263;
  b = 3;
  printf("0x%x\n", a ^ b);
  return 0;
}
```

Answers to the clicker questions

Question 1: A deque -- you can **push_front** the small ones and **push_back** the large ones, and the deque will be sorted. Each operation is *O(1)*.

I would give credit to a list -- you can perform the same operations on a list and have each operation be *O(1)*, but the list is slower than a deque (and less memory efficient as well).

You could do this with two vectors, but that is convoluted, so you'd need to explain how you're doing it.

Sets and maps will work, but they are slower *O(n log n)*.

Question 2: *O(n)*.

Question 3: 0x107. 256 + 7.

Question 4: 0x3. Here's the output of `bin/ba_helper` from the "Bits" lecture notes, with a bunch of leading zeroes and spaces trimmed:

```
Operator: AND
A:  263   0x0107  000100000111
B:    3   0x0003  000000000011
C:    3   0x0003  000000000011
```

Question 5: 0x104.

```
Operator: XOR
A:  263   0x0107  000100000111
B:    3   0x0003  000000000011
C:  260   0x0104  000100000100
```