Midterm Part 1B (35 Points)

1. (a) *Solution.* 30 billion.
   (b) *Solution.* 2 billion.
   (c) *Solution.* 1 million.
   (d) *Solution.* 3.6 million (will accept 3 or 4 million).
   (e) *Solution.* 1 million.
   (f) *Solution.* 20 times.
   (g) *Solution.* 10 million.
2. *Solution.* $[4, 7, 23, 9, 14, 2, 3, 12, 15, 17]$
3. (a) *Solution.* 1100
   (b) *Solution.* 011010
4. (a) *Solution.* $O(n)$
   (b) *Solution.* $O(\lg n)$
   (c) *Solution.* $O(\lg n)$

Midterm Part 2B (65 Points)

1. *Solution.* 00110, 11110, 10010, 10100, 10111.

2. *Solution.*

   (a) 2 Fenwick Trees: one for counting negatives, the other for counting zeros.

   (b) Change the corresponding entry in the negative Fenwick tree to 1.

   (c) Count the number of zeros and number of negatives in the range. If the number of zeros is positive, it is zero. Otherwise, it is positive iff the number of negatives is even.

3. *Solution.* 16

4. *Solution.*

```
int bs(int L, int R)
{
  if (L == R) return L;
  int M = (L+R)/2;
  return canComplete(M) ? bs(L,M) : bs(M+1,R);
}
int minStrength() { return bs(0,100000000); }
```

5. *Solution.*

```
long[][] cache = new long[10001][14]; //Filled with -1
long count(int n, int high)
{
  if (high < 0) return 0;
  if (n <= 0) return n==0?1:0;
  if (cache[n][high] != -1) return cache[n][high];
  return count(n,high-1) + count(n-(1<<high),high);
}
long count(int n) { return count(n,13); }
```