CS106X
Winter 2017

Chris Gregg

Section Handout #1 Solutions

If you have any questions about the solutions to the problems in this handout, feel free to reach out to your section leader, Aaron, or Chris for more information.

1. Mirror

```
void mirror(Grid<int> &grid) {
  for (int r = 0; r < grid.numRows(); r++) {
    for (int c = r + 1; c < grid.numCols(); c++) {  // start at r+1 rather
      int temp = grid[r][c];                         // than 0 to avoid
                                                     // double-swapping
      grid[r][c] = grid[c][r];
      grid[c][r] = temp;
    }
  }
}
```

2. Rotate Clockwise

```
void rotateClockwise90Degrees(Grid<int> &grid) {
  int size = grid.numRows();
  for (int layer = 0; layer < size / 2; layer++) { // move from outer layer to center
    int first = layer;
    int last = size - 1 - layer;

    // Go through the cells in a row/column to rotate
    for (int curr = first; curr < last; curr++) {
      int offset = curr - first;
      int top = grid[first][curr];                               // save top

      grid[first][curr] = grid[last - offset][first];         // left => top
      grid[last - offset][first] = grid[last][last - offset]; // bottom => left
      grid[last][last - offset] = grid[curr][last];           // right => bottom
      grid[curr][last] = top;                                 // top => right
    }
  }
}
```

3. Cumulative

```
void cumulative(Vector<int> &v) {
  for (int i = 1; i < v.size(); i++) {
    v[i] += v[i - 1];
  }
}
```

4. Stretch

```
void stretch(Vector<int> &v) {
  int size = v.size();

  for (int i = 0; i < size * 2; i += 2) {
    int n = v[i];
    v[i] = n / 2 + n % 2;
    v.insert(i + 1, n / 2);
  }
}
```

5. Big-Oh Notation

a. O(N)

b. O(N^2)

c. O(1)

d. O(N log N)

6. Oh? More Big-Oh?

a. O(N^2)

b. O(N^4)

c. O(N^2)

d. O(N)

7. Keith Numbers

```
bool findKeithSequence(Vector<int> &sequence, int n) {
  int sum = 0;
  int digits = n;
  int numDigits = 0;

  while (digits > 0) {
    int digit = digits % 10;
    sum += digit;
    sequence.insert(0, digit);
    digits /= 10;
    numDigits++;
  }

  while (sequence[sequence.size() - 1] < n) {
    sequence.add(sum);
    sum = sum - sequence[sequence.size() - numDigits - 1] + sum;
  }
  return sequence[sequence.size() - 1] == n;
}

void findKeithNumbers(int min, int max){
  for (int n = min; n <= max; n++) {
    Vector<int> sequence;
    if (findKeithSequence(sequence, n)) {
      // sequence ends in n? we have a Keith number
      cout << n << ": " << sequence << endl;
    }
  }
}
```