Chris Gregg
CS106B

Section #1
April 11, 2018

Solution to Section #1

Based on handouts by various current and past CS106B/X instructors and TAs.

1. Mirror

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

2. PlusSum

```
int plusSum(Grid<int> &grid, int row, int col) {
  int sum = 0;
  for (int c = col - 1; c <= col + 1; c++) {
    sum += grid[row][c];
  }
  for (int r = row - 1; r <= row + 1; r++) {
    sum += grid[r][col];
  }
  sum -= grid[row][col]; // subtract center because it was added twice
  return sum;
}
```

3. 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);
  }
}
```

4. Consecutive Duplicates

```
void removeConsecutiveDuplicates(Vector<int> &v) {
  for (int i = 0; i < v.size() - 1; i++) {
    if (v[i] == v[i + 1]) {
      v.remove(i + 1);
      i--;
    }
  }
}
```

5. Mystery

Stacks:	Output:
`{1, 2, 3, 4, 5, 6}`	`{6, 4, 2, 1, 3, 5}`
`{42, 3, 12, 15, 9, 71, 88}`	`{88, 12, 42, 3, 15, 9, 71}`
`{65, 30, 10, 20, 45, 55, 6, 1}`	`{6, 20, 10, 30, 65, 45, 55, 1}`

6. Duplicate Elements

```
void duplicateElements(Queue<int> &q) {
  int size = q.size();
  for (int i = 0; i < size; i++) {
    int n = q.dequeue();
    q.enqueue(n);
    q.enqueue(n);
  }
}
```

7. Split Stack

```
void splitStack(Stack<int> &stack) {
  Queue<int> queue;
  while (!stack.isEmpty()) {
    queue.enqueue(stack.pop());
  }
  int size = queue.size();
  for (int i = 0; i < size; i++) {
    int num = queue.dequeue();
    if (num >= 0) {
      queue.enqueue(num);
    } else {
      stack.push(num);
    }
  }
  while (!queue.isEmpty()) {
    stack.push(queue.dequeue());
  }
}
```

8. Big-O Analysis

a) $O(\text{N})$
b) $O(N^2)$
c) $O(1)$
d) $O(\text{NlogN})$
e) $O(N^2)$
f) $O(N^4)$
g) $O(N^2)$
h) $O(\text{N})$

9. 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)) {
        cout << n << ": " << sequence << endl;
    }
  }
}
```

10. Average in File

```
double averageValueInFile(string filename) {
  int count = 0;
  double sum = 0.0;
  ifstream input;
  openFile(input, filename);
  string line;
  while (getline(input, line)) {
    double val = stringToReal(line);
    count++;
    sum += val;
  }

  return 1.0 * sum / count;
}
```

11. Name Diamond

```
void nameDiamond(string s) {
  int len = (int)s.length(); // cast length to int to avoid warning
  // print top half of diamond
  for (int i = 1; i <= len; i++) {
    cout << s.substr(0, i) << endl;
  }

  // print bottom half of diamond
  for (int i = 1; i < len; i++) {
    // indent with spaces
    for (int j = 0; j < i; j++) {
      cout << " ";
    }
    cout << s.substr(i, len - i) << endl;
  }
}
```