CS106B Instructor: Cynthia Lee
Spring 2016 Practice Exam

PRACTICE MIDTERM EXAM #2 - SOLUTIONS

---

1. ADTs

```
void calculatePrefixes(Lexicon& english, Map<string,Set<string>>& prefixes) {
   for (string word : english) {
      for (int len = 1; len <= word.length(); len++) {
         prefixes[word.substr(0, len)] += word;
      }
   }
}

void printSuggestions(Map<string,Set<string>>& prefixes, string typing) {
   Set<string> suggestions;
   if (prefixes.containsKey(typing)) {
       suggestions = prefixes[typing];
       for (string s : suggestions) {
          cout << s << endl;
       }
   }
}
```

---

2. Linked Nodes [by Marty Stepp]

```
list2->next->next->next = list; // 4 -> 1
list->next = list2; // 1 -> 2
list = list2->next->next; // list -> 4
list2 = list2->next; // list2 -> 3
list2->next = NULL; // 3 /
list->next->next->next = NULL; // 2 /
```

---

3. Recursion

```
void sum21(Vector<int>& availableCards, Vector<Vector<int>>& waysToSum) {
    Vector<int> hand;
    sum21(availableCards, waysToSum, hand, 0);
}

void sum21(Vector<int>& availableCards, Vector<Vector<int>>& waysToSum,
```

```
            Vector<int>& hand, int sum) {
    if (sum == 21) {
        waysToSum += hand;
        return;
    }
    if (sum > 21 || availableCards.size() == 0) {
        return;
    }
    int card = availableCards[0];
    availableCards.remove(0);
    // try without this card
    sum21(availableCards, waysToSum, hand, sum);
    hand.add(card);
    // try with this card
    sum21(availableCards, waysToSum, hand, sum + card);
    // special case: try with this card as 11-pt Ace
    if (card == 1) sum21(availableCards, waysToSum, hand, sum + 11);
    hand.remove(hand.size() - 1);
    availableCards.insert(0, card);
}
```

---

4. **Classes**

```
int ArrayList::maxCount() const {
   if (mysize == 0) {
      return 0;
   }
   int max = 1;
   int count = 1;
   for (int i = 1; i < mysize; i++) {
      if (elements[i] == elements[i - 1]) {
         count++;
         if (count > max) {
            max = count;
         }
      } else {
         count = 1;
      }
   }
   return max;
}
```

---

5. **Short answer**

(a) Vector, Grid
(b) False—the address-of & operator can find the address of any named variable, and the address can be stored in a pointer variable
(c) False—that is bad style (arm's length recursion)
(d) O(N)
(e) O(1)
(f) O(N^2)
(g) (1) Address-of operator takes the address of any named variable. (2) Pass by reference lets you pass an argument by reference to a function.
(h) (1) For efficiency for a large data structure. (2) If you want to effectively return more than one value, you can use reference parameters as effective return values.