Hacking a Google Interview Practice Questions – Person A

Question: Substring

Write a program to determine whether an input string x is a substring of another input string y. (For example, "bat" is a substring of "abate", but not of "beat".) You may use any language you like.

Sample Answer (in C++):

```
bool hasSubstring(const char *str, const char *find) {
    if (str[0] == '\0' && find[0] == '\0')
        return true;

    for(int i = 0; str[i] != '\0'; i++) {
        bool foundNonMatch = false;
        for(int j = 0; find[j] != '\0'; j++) {
            if (str[i + j] != find[j]) {
                foundNonMatch = true;
                break;
            }
        }
        if (!foundNonMatch)
            return true;
    }
    return false;
}
```

Question: Text Editor

Describe a design for a text editor. Describe the classes, interfaces, and so on that you would use and how you would organize them.

Answer: There are so many possible answers to this problem that it would be difficult to say that one answer is the best. Look to make sure that they make classes to set up a text editor (classes for the GUI, formatting, saving/loading files, handling input, etc.). Using inheritance (subclassing in object-oriented programming) where it makes sense is also good for reusability and extendability. Using design patters (such as Model-View-Controller, Listener/Observer, or the Singleton pattern) is also a good thing. The main point is for them to get used to thinking about how they would design a system. Most importantly, they need to think about simplicity, reusability, and extendability in their design.

A text editor design question is slightly different from other design questions in that programmers often have strong feelings about how a text editor should work. Programmers often want the ability to greatly modify the behavior of their editor and want to be able to write extensions that add functionality to it. The major text editors used by programmers today, such as Emacs, Vim, Eclipse, and Visual Studio have this ability. A discussion about how their text editor would accomplish this (especially with how the design would include a place for extensions and how input would be handled) would be good.

Question: Axis-Aligned Rectangles

Describe an algorithm that takes an unsorted array of axis-aligned rectangles and returns any pair of rectangles that overlaps, if there is such a pair. Axis-aligned means that all the rectangle sides are either parallel or perpendicular to the x- and y-axis. You can assume that each rectangle object has two variables in it: the x-y coordinates of the upper-left corner and the bottom-right corner.

Good Answer: Create a sorted array of the x coordinates of the left and right edges of the rectangles. Then, use a "scanline" to move from left to right through the rectangles. Keep a binary search tree containing the y coordinates of the top and bottom edges of the rectangles that overlap the scanline. For each element of the array, check whether it is a left or right edge. If it is a right edge, remove the corresponding top and bottom edges from the BST. If it is a left edge, search the BST for rectangles that overlap the current rectangle; if there is one, return the overlap. Then, add the y coordinates of the top and bottom edges of the rectangle to the BST. The search takes O(n log n) time, since it takes O(n log n) time to sort the rectangles and each of the 2n iterations takes O(log n) time.

Question: Doubly Linked List

Write a function to remove a single occurrence of an integer from a doubly linked list if it is present. You may use any language you like.

Sample Answer (in Java):

```
void remove(Node head, int value) {
    Node cur = head;
    while (cur != null) {
        if (cur.value == value) {
            if (curr.prev != null)
                cur.prev.next = cur.next;
            if (cur.next != null)
                cur.next.prev = cur.prev;
            break;
        }
        cur = cur.next;
```

```
        }
}
```

Question: Minimum Stack

Describe a stack data structure that supports "push", "pop", and "find minimum" operations. "Find minimum" returns the smallest element in the stack.

Good Answer: Store two stacks, one of which contains all of the items in the stack and one of which is a stack of minima. To push an element, push it onto the first stack. Check whether it is smaller than the top item on the second stack; if so, push it onto the second stack. To pop an item, pop it from the first stack. If it is the top element of the second stack, pop it from the second stack. To find the minimum element, simply return the element on the top of the second stack. Each operation takes O(1) time.

Question: Hash Tables

Describe how a hash table works.

Answer: You can refer to handout 2 for a description of hash tables.

Question: Coin Flipping and Die Rolls

Describe an algorithm to output a die roll (a random number from 1 to 6), given a function that outputs a coin toss (a random number from 1 to 2). Each possible outcome should be equally likely.

Sample Answer: Flip the coin three times, and use the three coin flips as the bits of a three-bit number. If the number is in the range 1 to 6, output the number. Otherwise, repeat. Note that many other answers are possible.

Question: Target Sum

Given an integer x and an unsorted array of integers, describe an algorithm to determine whether two of the numbers add up to x. (In this case, say that the interviewer hates hash tables.)

Good Answer: Sort the array. Then, keep track of two pointers in the array, one at the beginning and one at the end. Whenever the sum of the current two integers is less than x, move the first pointer forwards, and whenever the sum is greater than x, move the second pointer backwards. If you cannot find two numbers that add to x before one of the pointers meet, then there is no pair of integers that sum to x. This solution takes O(n log n) time because we sort the numbers.

Another Good Answer: Create a binary search tree containing x minus each element in the array. Then, check whether any element of the array appears in the BST. It takes O(n log n) time to create a binary search tree from an array, since it takes O(log n) time to insert something into a BST, and it takes O(n log n) time to see if any element in an array is in a BST, since the lookup time for each element in the array takes O(log n). Therefore step one takes O(n log n) time and step two takes O(n log n) time, so our total running time is O(n log n).

Question: Debugging

Describe a good strategy to find a bug in a program.

Answer: This question has many possible answers, and is the sort of open-ended question that interviewers occasionally ask. A good answer to this question might include identifying the portion of the program in which the bug appears to be occurring based on its behavior, as well as using breakpoints and a stepper to step through the program. Any answers that involve thinking about possible sources of the problem and finding ways to limit the search scope of the bug are good answers.