1 Strategy

Table 1: ***Test table***

Date	Todo	Outcome
Sept 30	Basics, list of topics to focus on	
Oct. 01	2 hrs of top-coder	
	Binary search	
	BST	
Oct. 02	1 hr of topcoder	
	Arrays and Lists	
Oct. 03	1 hr of careercup	
	Queueus, Stacks	
Oct. 04	1 hr of careercup.	
	Heap, priority heaps	
Oct. 05	1 topcoder and 1 careercup	
	Trees	
	String manipulations	
Oct. 06	Search trees	
Oct. 07	Secure location for the test	
	Test setup (mult machines and screen)	
Oct. 08	Day of the	

2 References

- Big-O Cheat Sheet - a must have for any interview
- Sample Interview Questions
- Intro to programming in C
- Guide to C
- Programming concepts
- Java tutorial - concepts
- java functions
- Introduction to Object Oriented Programming Concepts (OOP) and More
- OOP Brush-up
- Data Structures
- From Java code to Java heap
- Data Structures and java.util and How to implement common data structures (List, Stack, Map) in plain Java
- Fibonacci - Java
- Using Stacks: Reversing the elements of a string

2.1 Media

- Stanford Seminar - Google's Steve Yegge on GROK
- Dynamic Languages Strike Back

3 Review of Standard Libraries

4 Common Data Structures

Arrays, Linked Lists, Binary Search, Binary Search Trees, Hash Tables, Sets, and Graphs Fibonacci

- **Singly linked lists**
- **Doubly linked lists**
- **Heap**
- **Stacks**

  Reversing a string using a stack
- **Queueus**
- **Hashtables**
- **Trees and Binary Trees**
- **Recursion**
- **Graphs** A graph data structure consists of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices [Reference].
- **BST**

  Binary Search Trees

  Details of the BST java implementation

  Binary search algorithm Finds the position of a specified input value (the search "key") within an array sorted by key value. The binary search is the standard algorithm for searching through a sorted sequence.

  Binary search is a logarithmic algorithm and executes in $O(logN)$ time.

```
int binary_search(int A[], int key, int imin, int imax)
{
// test if array is empty
if (imax < imin)
  // set is empty, so return value showing not found
  return KEY_NOT_FOUND;
else
  {
    // calculate midpoint to cut set in half
    int imid = midpoint(imin, imax);

    // three-way comparison
    if (A[imid] > key)
      // key is in lower subset
      return binary_search(A, key, imin, imid-1);
    else if (A[imid] < key)
      // key is in upper subset
      return binary_search(A, key, imid+1, imax);
```

```
    else
      // key has been found
      return imid;
  }
}
```

- **Arrays and Strings** An object consisting of a seq of elements; can be created dyn at runtime searching/hashtables

5 OO Object Oriented must knows

- **class, object (and the difference between the two)** A class is template for objects and describes object behavior; a class is an abstraction; A class is a blueprint or prototype from which objects are created.

  An object is an instance or member of a class; objects has defined states; it describes a relationship

- **instantiation** The act of creating an object is called instantiation. Create an instance of an class in the form of an object.

- **method (as opposed to, say, a C function)** Concept of an " object" which is a data structure (abstract data type) encapsulated with a set of routines, called " methods", which operate on the data. Operations on the data can only be performed via these methods,

- **virtual method, pure virtual method** Virtual function or virtual method is a function or method whose behavior can be overridden within an inheriting class by a function with the same signature. This concept is a very important part of the polymorphism Example: class Animal; class Fish inherits from Animal, but overrides eat() so that Fish or Wolf can have the appropriate methods see here.

- **final classes** final classes can have no subclasses

- **class/static method** There are two types of methods in Java, called class methods and object methods. Class methods are identified by the keyword static in the first line. Any method that does not have the keyword static is an object method. But, why?

- **static/class initializer** Programming constructs which perform initialization are typically called initializers and initializer lists. Initialization is distinct from (and preceded by) declaration, although the two can sometimes be conflated in practice.

- **constructor** a special type of subroutine called to create an object. In C++: very time an instance of a class is created the constructor method is called. The constructor has the same name as the class and it doesn't return any type, while the destructor's name it's defined in the same way, but with a ' ' in front

- **destructor/finalizer** Destructors are cleanup methods invoked deterministically and finalizers when the garbage collector tells them. Destructors are called to destruct instances of classes.

  In Java there is no equivalent of a destructor

- **superclass or base class** Common interface and foundational functionality Example: class Bycicle; subclass: MoutainBike, RoadBike

- **subclass or derived class** As part of inheritance, we have subclassing, a class that inherits from a superclass

- **inheritance** Inheriting members of an existing class (base class), a relationship to existing class members; inheriting data and functions from another class

- **encapsulation** mechanism for restricting access to some of the object's components An informaition hiding mechanism by defining methods to be Public or Private (in Java) Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the inter-dependencies between software components.
- **multiple inheritance (and give an example)** Deriving directly from more than one class Trivia: multiple inheritance is not supported like Java
- **delegation/forwarding** Delegation is an alternative to inheritance for reusing code among multiple classes. Inheritance uses the IS-A relationship for reuse; delegation uses the HAS-A reference relationship to do the same.

  The delegate is acting on behalf of (performing some of the function of) the original object. In Objective-C, protocols describe the methods that delegates can or must implement in order to perform the activities they've been delegated to do.
- **composition/aggregation** composition gives us a 'part-of' relationship. A "owns" B = Composition : B has no meaning or purpose in the system without A A "uses" B = Aggregation : B exists independently (conceptually) from A Example: A Text Editor owns a Buffer (composition). A Text Editor uses a File (aggregation). When the Text Editor is closed, the Buffer is destroyed but the File itself is not destroyed.
- **abstract class** An abstract class is one that is not used to construct objects, but only as a basis for making subclasses. An abstract class exists only to express the common properties of all its subclasses.
- **interface/protocol (and different from abstract class)** An interface is a description of the actions that an object can do an Interface is a description of all functions that an object must have in order to be an "X".
- **method overriding** Method overriding is a language feature that allows a subclass to override a specific implementation of a method that is already provided by one of its super-classes.
- **method overloading (and difference from overriding)** The method overloading is the ability to define several methods all with the same name. (we do this a lot in Java)
- **polymorphism (without resorting to examples)** In OOP the polymorphisms is achieved by using many different techniques named method overloading, operator overloading and method overriding,
- **is-a versus has-a relationships (with examples)** As we know, inheritance gives us an 'is-a' relationship.
- **method signatures (what's included in one)** A method is commonly identified by its unique method signature, which usually includes the method name, and the number, types and order of its parameters. A Java static method can provide only one return value, of the type declared in the method signature.
- **method visibility (e.g. public/private/other)** For fields and methods, we have, in addition to public, three more visibility modifiers: internal, protected, and private.

6 More on Concepts

Real-world objects contain state and behavior. A software object's state is stored in fields. A software object's behavior is exposed through methods. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data encapsulation. A blueprint for a software object is called a class. Common behavior can be defined in a superclass and inherited into a subclass using the extends keyword. A collection of methods with no implementation is called an interface. A namespace that organizes classes and interfaces by functionality is called a package. The term API stands for Application Programming Interface.

7 Algorithms

7.1 Sorting

- **Bubble sort**
- **Selection sort**
- **Merge-sort**
- **Quick sort**
- **Radix-sort**
- **Backtracking** Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. It is useless, for example, for locating a given value in an unordered table [Reference].
- **Divide and Conquer** This technique is the basis of efficient algorithms for all kinds of problems, such as sorting (e.g., quicksort, merge sort), multiplying large numbers (e.g. Karatsuba), syntactic analysis (e.g., top-down parsers), and computing the discrete Fourier transform (FFTs).[Reference]

8 Examples

8.1 Fibonacci

```
public static long fib(int n) {
       if (n <= 1) return n;
       else return fib(n-1) + fib(n-2);
   }
```

8.2 Bit structures

Write a program to store logical values in bit using structure.

Solution

```
# include <stdio.h>
# include <conio.h>

 main ()
 {

struct bits
{
int state :1;
};

struct bits a;
a.state=13>5;
clrscr();
if (a.state==0) printf ("False");
else printf ("True : %d",5>1);
}

OUTPUT
True: 1
```

Explanation In the structure bits, the number of bits given is to be used by the member of structure to store values. The value returned by the expression 13>5 are stored in variable state that stores the value in one bit. The if statement checks the value and the appropriate message is displayed.

8.3 Fibonacci

A naive implementation, an optimized version should use memoization to deal with runtime.

```
public class prob2 {
       // naive recursive implementation
       public static long fib(int n) {
       if (n <= 1) return n;
       else return fib(n-1) + fib(n-2);
   }

       /* Each new term in the Fibonacci sequence is generated by adding
        * the previous two terms. By starting with 1 and 2, the first 10
        * terms will be:

              1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

        * By considering the terms in the Fibonacci sequence whose values
        * do not exceed four million, find the sum of the even-valued
        * terms.
        */

       public static void main(String[] args) {
              long curr_fib = 0L;
              long even_sum = 0L;
              int n =2;

              while(curr_fib <= 4000000) {
                     if ( (curr_fib = fib(n)) % 2 == 0)
                            even_sum += curr_fib;
                     n++;
              }
              System.out.println("Sum: "+even_sum);
       }
}
```

8.4 Java Literal types range

Reference

```
char testChar = 01;
byte testByte = -128;
int testInt = -2147483648;
short testShort = -32768;
long testLong = 9223372036854775807L;
float testFoat;
double testDouble = 4.940656458412;
boolean testBool = true;
```

8.5 Working with pointers

9 Strengths and Weaknesses of C, C++, Java, Python, Objective-C

- java.pdf
- General comparison
- Comparison of Java and C
- Python compared to others (Java, C++

10 Referenced Books

1. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (Jul 31, 2009)
2. Algorithms (4th Edition) [Hardcover] Robert Sedgewick (Author), Kevin Wayne (Author)
3. Cracking the Coding Interview: 150 Programming Questions and Solutions [Paperback] Gayle Laakmann McDowell (Author)