1 List’em all!

List all the asymptotic runtimes from quickest to slowest.
\( \theta(n^2), \theta(n^{0.5}), \theta(\log n), \theta(3^n), \theta(c), \theta(n^n), \theta(n^{n!}), \theta(n\log n), \theta(n!), \theta(n^n), \theta(2^n) \)

2 What’s that runtime?

For each of the methods below, please specify the runtime in BigO, Big\( \Theta \) or Big\( \Omega \) Notation. Please give the tightest bound possible.

```java
public static void f(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            linear(n); // runs in linear time with respect to input
        }
    }
}
```

```java
public static void g(int n) {
    if (n < 1) return;
    for (int i = 0; i < n; i++) {
        linear(100);
    }
    g(n/2);
    g(n/2);
}
```

```java
public static void h(int n) {
    Random generator = new Random();
    for (int i = 0; i < n; i++) {
        if (generator.nextBoolean()) {
            break;
        }
    }
}
```
3 How fast?

Given an IntList of length N, provide the runtime bound for each operation. Recall that IntList is the naive linked list implementation from class.

```java
public class IntList {
    int item;
    IntList next;
}
```

<table>
<thead>
<tr>
<th>Operations</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>size()</td>
<td></td>
</tr>
<tr>
<td>get(int index)</td>
<td></td>
</tr>
<tr>
<td>addFirst(E e)</td>
<td></td>
</tr>
<tr>
<td>addLast(E e)</td>
<td></td>
</tr>
<tr>
<td>remove(int index)</td>
<td></td>
</tr>
<tr>
<td>remove(Node n)</td>
<td></td>
</tr>
</tbody>
</table>

CS 61B, Summer 2020, Worksheet 4
4 The ABCs of OOP

Indicate what each line the main program in class D would print, if the line prints anything. If any lines error out, identify the errors as compile-time or runtime errors and cross out the corresponding lines.

```java
class A {
    public void x() { System.out.println("Ax"); }
    public void y(A z) { System.out.println("Ay"); }
}

class B extends A {
    public void y() { System.out.println("By"); }
    public void y(B z) { System.out.println("Byz"); }
}

class C extends A {
    public void x() { System.out.println("Cx"); }
}

class D {
    public static void main(String[] args) {
        A e = new B();
        A f = new C();
        B g = new A();
        B h = new C();
        C i = (C) new A();
        B j = (A) new C();
        B k = (B) e;

        f.x();
        e.x();
        e.y();
        (B) e.y();
        ((B) e).y();
        e.y(e);
        e.y(f);
    }
}
```
5 Classy Cats

Look at the Animal class defined below.

```java
public class Animal {
    protected String name, noise;
    protected int age;

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }

    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }

    public String greet() {
        return name + " : " + makeNoise();
    }
}
```

(a) Given the Animal class, fill in the definition of the Cat class so that it makes a "Meow!" noise when greet() is called. Assume this noise is all caps for kittens, i.e. Cats that are less than 2 years old.

```java
public class Cat extends Animal {
}
```
(b) “Animal” is an extremely broad classification, so it doesn’t really make sense to have it be a class. Look at the new definition of the Animal class below.

```java
public abstract class Animal {
    protected String name;
    protected String noise = "Huh?";
    protected int age;

    public String makeNoise() {
        if (age < 2) {
            return noise.toUpperCase();
        }
        return noise;
    }

    public String greet() {
        return name + ": " + makeNoise();
    }

    public abstract void shout();
    abstract void count(int x);
}
```

Fill out the Cat class again below to allow it to be compatible with Animal (which is now an abstract class) and its two new methods.

```java
public class Cat extends Animal {
    public Cat() {
        this.name = "Kitty";
        this.age = 1;
        this.noise = "Meow!";
    }

    public Cat(String name, int age) {
        this();
        this.name = name;
        this.age = age;
    }

    @Override
    public void shout() {
        System.out.println(noise.toUpperCase());
    }

    @Override
    public void count(int x) {
        for (int i = 0; i < x; i++) {
            System.out.println(noise);
        }
    }
}
```
System.out.println(makeNoise());

}