Q1. /** = “this rhino or one of its ancestors has name n.” */

```java
public boolean hasName(String n) {
    if (String.equals(n)) return true;
    return (father == null ? false : father.hasName(n)) ||
           (mother == null ? false : mother.hasName(n));
}
```

Q2. (a) Output: the message “Exception!!!”

(b) ```java
public class BadNumberException extends Exception {
    /** Constructor: instance with empty string for a detail message */
    public BadNumberException() {
        super();
    }
    /** Constructor: instance with detail message d */
    public BadNumberException(String d) {
        super(d);
    }
}
```

(c) /** = greatest common divisor of x and y.
Throw a BadNumberException if x<=0 or y<0. */

```java
public static int GCD(int x, int y) throws BadNumberException {
    if (x <= 0 || y <= 0) {
        throw new BadNumberException(“x and y have to be positive integers”);
    }
    int b= x; int c= y;
    /* inv: gcd(x, y) = gcd(b, c), b > 0, and c > 0 */
    while (b != c) {
        if (b < c) c= c – b;
        else b= b – c;
    }
    return b;
}
```

Q3. (a) // inv: each row 0..k-1 of sq sums to sum.
for (int k= 0; k < sq.length; k = k + 1) {
    // Return false if row k does not sum to sum.
    int rowsum= 0;
    for (int j= 0; j < sq.length; j = j + 1) {
        rowsum= rowsum + sq[k][j];
    }
    if (rowsum != sum)
        return false;
}
/* post: each row 0..sq.length-1 sums to sum */

Q4. (a) Code in a construct can reference any of the names declared in that construct as well as names that appear in enclosing constructs (unless a name is declared twice, in which case the closer one prevails).

(b) Within an object: this refers to the object itself, while super refers to the object but only the partitions for the superclass and above. Also, “this(...)” can be used to call another constructor in this object and “super(...)” can be used to call a constructor in the superclass partition of the object.

(c) To override a method is to redeclare an inherited method in a class. this.m(...) or m(…)

Q5. (a) /** = an integer j that satisfies
b[p..j] <= x < b[j+1..q-1]
Precondition: b[p..q-1] is sorted */

```java
public static int bsearch(int[] b, int x, int p, int q) {
    int j= p-1;
    int k= q;
    // invariant: b[p..] <= x < b[k..]
    while (j+1 != k) {
        int e= (j+k) / 2;
        if (b[e] <= x) j= e;
        else k= e;
    }
    return j;
}
```