1. (10 points): Consider the class below. Notice that `shrubbery()` is not static. Precisely what lines of code could be placed within `ni()` in order to call `shrubbery()`?

```java
public class A {
    public void shrubbery() {
        System.out.println("Bring me a shrubbery.");
    } // shrubbery()

    public static void ni() {
        // ni()
    }

    public static void main(String[] args) {
        ni();
    } // main()
}
```
2. (10 points): What output is generated by the following program? Don’t worry about spacing, but show the right output on the right line(s).

```java
public class Trace2 {
    protected int a;
    protected int b;

    public Trace2(int x) {
        a = x;
        b = a + x;
    } // constructor

    public Trace2(int x, int y) {
        a = y;
        b = x;
    } // constructor

    public int f(int z) {
        return a * (b - z);
    } // f()

    public int f() {
        return a + b;
    } // f()

    public String toString() {
        return "a = " + a + " ; b = " + b;
    } // toString()

    public static void main(String[] args) {
        Trace2 t = new Trace2(3, 5);
        Trace2 u = new Trace2(7);
        System.out.println(u.f(2));
        System.out.println(t.f());
        System.out.println(t.toString() + " : " + u.toString());
    } // main()
} // class Trace2
```
3. **(10 points)**: Write a class that extends Trace2 from the previous problem and overrides both versions of \( f() \) so that the one that takes \( z \) as a parameter returns the smallest of \( a, b, \) and \( z \), and the \( f() \) that takes no parameter returns the smallest of \( a \) and \( b \). For simplicity, omit the constructors.
4. (10 points): Show the output produced by the following program when `main()` in class `SubFin` runs.

class Fin {

    protected int[] data;
    public Fin(int n) {
        data = new int[n];
        for (int index = 0; index < n; index++)
            data[index] = 1;
    } // constructor

    public void twist(int spot) {
        for (int index = 1; index < spot; index++)
            data[index] = data[index] + data[index - 1];
    } // twist()

    public void display() {
        for (int value: data)
            System.out.println(value);
    } // display()

} // class Fin

class SubFin extends Fin {

    public SubFin(int n) {
        super(n);
        int mid = n / 2;
        data[mid] = 0;
    } // constructor

    public void twist(int spot) {
        for (int index = spot - 1; index < data.length; index++)
            data[index] = data[index] + data[spot];
        super.twist(spot);
    } // twist

    public static void main(String[] args) {
        Fin f = new SubFin(6);
        f.twist(4);
        f.display();
    } // main()
} // class SubFin