```
 1:
 2: /**
 3:  * This class represents a U.S. telephone number. It supports three types of
 4:  * constructors -- one that accepts three numbers, representing area code,
 5:  * exchange, and extension; another that accepts two integers, representing a
 6:  * number within your local area code; and a third constructor that accepts
 7:  * a string of letters and numbers that represent the number
 8:  * (e.g., "900-410-TIME"). There is also a method that determines if the
 9:  * number provided is toll-free (such numbers have area codes of 800, 866, 877,
10:  * 880, 881, 882, or 888).
11:  *
12:  * @author 1828799
13:  */
14: public class Telephone
15: {
16:
17:    // Instance Variable
18:    private String _phoneNumber;
19:    private int _areaCode;
20:
21:
22:    /**
23:     * A constructor that accepts three numbers; an area code, exchange,
24:     * and extension
25:     * @param areaCode the area code of the number (first 3)
26:     * @param exchange the exchange of the number (middle 3)
27:     * @param ext the number's extension (last 4)
28:     */
29:    public Telephone(int areaCode, int exchange, int ext)
30:    {
31:       _areaCode = areaCode;
32:       _phoneNumber = Integer.toString(areaCode) + "-"
33:             + Integer.toString(exchange) + "-" + Integer.toString(ext);
34:    }
35:
36:
37:    /**
38:     * A constructor that accepts two integers that represent a number within
39:     * the local area code
40:     * @param exchange the exchange of the number (middle 3)
41:     * @param ext the extension of the number (last 4)
42:     */
43:    public Telephone(int exchange, int ext)
44:    {
45:       _areaCode = 503;
46:       _phoneNumber = Integer.toString(_areaCode) + "-"
47:             + Integer.toString(exchange) + "-" + Integer.toString(ext);
48:    }
49:
50:
51:    /**
52:     * A constructor that accepts a string of letters and numbers that
53:     * represent the number (e.g., "900-410-TIME")
54:     * @param phone the string representation of the number
55:     */
56:    public Telephone(String phone)
57:    {
58:       _areaCode = Integer.valueOf(phone.substring(0, 2));
59:       _phoneNumber = phone;
60:    }
61:
62:
63:    /**
64:     * This method determines if the number is toll free or not
65:     * @param areaCode the area code
```

```
66:     * @return boolean of if the area code is toll free or not
67:     */
68:    public boolean tollFree()
69:    {
70:       boolean free = false;
71:
72:       if(_areaCode == 888 || _areaCode == 866 || _areaCode == 877
73:             || _areaCode == 880 || _areaCode == 881 || _areaCode == 882
74:             || _areaCode == 888)
75:       {
76:          free = true;
77:       }
78:
79:       return free;
80:    }
81:
82:    /**
83:     * Converts _phoneNumber to a string
84:     * @return _phoneNumber the phone number
85:     * @override toString
86:     */
87:    public String toString()
88:    {
89:       return _phoneNumber;
90:    }
91:
92: }
93:
94:
```

Student: 1828799
Problem 1.2

a, d, e, f, h