

CS 106A Winter 2017 Final Solutions

1a.

```
public void run() {  
    for(int i = 100; i >= 0; i -= 5) {  
        println(i);  
    }  
}
```

1b.

```
public void printKeys(HashMap<String, String> map) {  
    for(String key : map.keySet()) {  
        println(key);  
    }  
}
```

1c.

```
private char largestLetter(String str) {  
    char largest = 'a';  
    for(int i = 0; i < str.length(); i++) {  
        char curr = str.charAt(i);  
        if(curr > largest) {  
            largest = curr;  
        }  
    }  
    return largest;  
}
```

1d.

```
flowers[0] = 2  
flowers[1] = 5
```

2.

```
public class EReader extends ConsoleProgram {

    private int currPage = 1;

    public void init() {
        add(new JButton("Previous"), SOUTH);
        add(new JButton("Next"), SOUTH);
        addActionListeners();
    }

    public void run() {
        displayCurrPage();
    }

    public void actionPerformed(ActionEvent e) {
        String cmd = e.getActionCommand();
        if(e.equals("Previous")) {
            if(currPage != 1) currPage--;
        } else {
            if(currPage != 100) currPage++;
        }
        displayCurrPage();
    }

    private void displayCurrPage(){
        printFile("page" + currPage + ".txt");
    }

}
```

3.

```
public class ChangingMindsets extends GraphicsProgram {

    public void run() {
        try {
            BufferedReader rd
                = new BufferedReader(new FileReader("2015.txt"));
            while(true) {
                String line = rd.readLine();
                if(line == null) return;
                String[] parts = line.split(" ");
                double wealth = Double.parseDouble(parts[1]);
                double health = Double.parseDouble(parts[2]);
                double pop = Double.parseDouble(parts[3]);
                double x = wealth * getWidth();
                double y = health * getHeight();
                double r = Math.sqrt(pop / Math.PI);
                drawCircle(x, y, r);
            }
        } catch(IOException e) {
            e.printStackTrace();
        }
    }

    private void drawCircle(double x, double y, double r) {
        GOval circle = new GOval(2 * r, 2 * r);
        add(circle, x - r, y - r);
    }
}
```

4.

```
private String[] makeAscii(GImage img) {
    double[][] brightness = img.getPixelBrightness();
    String[] lines = new String[brightness.length];
    for(int r = 0; r < lines.length; r++) {
        String line = "";
        for(int c = 0; c < brightness[0].length; c++) {
            double v = brightness[r][c];
            if(v > 0.66) {
                line += ' ';
            } else if (v > 0.66) {
                line += '1';
            } else {
                line += '0';
            }
        }
        lines[r] = line;
    }
    return lines;
}
```

5.

```
public class GoogleImages extends GraphicsProgram {
    private static final int ROW_HEIGHT = 300;
    private static final int GAP = 20;
    private static final int TEXT_FIELD_SIZE = 20;

    private JTextField qField
        = new JTextField(TEXT_FIELD_SIZE);

    public void init() {
        add(qField, SOUTH);
        add(new JButton("Search"), SOUTH);
        addActionListeners();
    }

    public void actionPerformed(ActionEvent e) {
        String query = qField.getText();
        ArrayList<GImage> results = getSearchResults(query);

        int index = 0;
        int row = 0;
        int currX = GAP;
        int currY = GAP;

        while(row < 3) {
            GImage img = results.get(index);
            double ratio = img.getWidth() / img.getHeight();
            double width = ROW_HEIGHT * ratio;
            if(currX + width < getWidth()) {
                add(img, currX, currY);
                currX += width + GAP;
                index++;
            } else {
                row++;
                currX = GAP;
                currY += ROW_HEIGHT + GAP;
            }
        }
    }
}
```

6a.

// Problem 6a: Note Class (12 points)

```
public class Note {  
  
    private String name;  
    private int duration;  
  
    // the constructor  
    public Note(String name, int duration) {  
        this.name = name;  
        this.duration = duration;  
    }  
  
    // returns the note's name  
    public String getName() {  
        return name;  
    }  
  
    // returns the note's duration  
    public int getDuration() {  
        return duration  
    }  
}
```

6b.

// Problem 6b: Song Class (18 points)

```
public class Song {

    private ArrayList<Note> notes;
    private int length = 0;

    // the constructor
    public Song() {
        notes = new ArrayList<Note>();
    }

    // appends a new note to the song
    public void addNote(Note newNote) {
        notes.add(newNote);
        length += newNote.getDuration();
    }

    // returns the total length of the song (in number of beats)
    // note that number of beats does not equal number of notes.
    public int getSongLength() {
        return length;
    }

    //returns the note name this many beats into the song.
    public String getNoteAtTime(int time) {
        int currTime = 0;
        for(Note n : notes) {
            currTime += n.getDuration();
            if(currTime > time) {
                return n.getName();
            }
        }
        return "";
    }
}
```

7.

```
private HashMap<String, double[]> join (
    HashMap<String, Double> a,
    HashMap<String, Double> b) {

    HashMap<String, double[]> joined = new HashMap<String, double[]>();

    for(String key : a.keySet()) {
        if(b.containsKey(key)) {
            double[] value = new double[2];
            value[0] = a.get(key);
            value[1] = b.get(key);
            joined.put(key, value);
        }
    }

    return joined;
}
```