1.

```
#!/bin/bash

# Pull names from a Linux text file
function getNames() {
       grep -Eo '\w+[\.]' $1 | tr -d '.'
}
```

2.

```
#!/bin/bash

# Pull unique names from a Linux text file
function getUnique() {
      grep -Eo '\w+[\.]' $1 | tr -d '.' | sort --unique
}
```

3.

```
#!/bin/bash

# Join every two lines together in a Linux text file
function joinByTwo() {
      grep . $1 | paste -s -d '\t\n'
}
```

4.

```
#!/bin/bash

# Pull logs with FIX tag 52 from a Linux text file
function getLogs() {
      grep . $1 | paste -s -d '\t\n' | grep -E '([0][5][2])+'
}
```

5.

Java Solution (FixLogFinder.java)

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class FixLogFinder {
	public void getLogs(String fileName, String fixTag) {
	File file = new File(fileName);

	try {
		Scanner scanner = new Scanner(file);

		while (scanner.hasNextLine()) {
			String line = scanner.nextLine();
			if (line.contains(fixTag)) {
				System.out.print(line);
				// Since the rest of the log is on the next line,
				// we advance the scanner
				// forward by one line and print it out.
				line = scanner.nextLine();
				System.out.print(" " + line + "\n");
			}
		}

	} catch(FileNotFoundException e) {
		System.out.println("File " + fileName + " was not found.");
		e.printStackTrace();
	}
}

public static void main(String[] args) {
	FixLogFinder logFinder = new FixLogFinder();
	logFinder.getLogs("test4.txt", "052");
	}
}
```

6.

```
find ~/tmp -type f -mtime +13 -delete
```

7.

Java Solution (LogFinder.java)

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class LogFinder {
	public void getLogs(String fileName) {
	File file = new File(fileName);

	try {
		Scanner scanner = new Scanner(file);

		while (scanner.hasNextLine()) {
			String line = scanner.nextLine();
			if (line.toUpperCase().contains("[WARN]") ||
line.toUpperCase().contains("[ERROR]")
				|| line.toUpperCase().contains("[FATAL]")) {
			System.out.println(line);
			}
		}
	} catch(FileNotFoundException e) {
		System.out.println("File " + fileName + " was not found.");
		e.printStackTrace();
	}
	}

	public static void main(String[] args) {
		LogFinder logFinder = new LogFinder();
		logFinder.getLogs("gelber_ops.log");
	}
}
```

8.

To solve this problem, I would first break it down into parts to make it more manageable and easier to see what we need to do. The problem boils down to three major components:

1. We need to iterate through each .html file in a directory.
2. For each file, we have to check if it contains one of two phone number formats.
3. If the file has a phone number, we need to obtain its filepath.

From there, we can come up with a solution for each of the steps.

1. We can use a for loop to iterate through the files.
2. We can use regex matching to confirm that a file has a phone number format we are looking for.
3. We can get the path of the file while it is the current one in the loop.

This general idea can be adapted to whatever language would be best for working with a Unix directory tree. I chose to use Python3 for its speed and simplicity, and came up with this sample script:

(phonenumbers.py)

```
import os
import re

# regex matches phone numbers with formats xxx-xxx-xxxx or (xxx)
xxx-xxxx
r = re.compile("([(][0-9]{3}[)][
][0-9]{3}[-][0-9]{4})|([0-9]{3}[-][0-9]{3}[-][0-9]{4})")

directory = "test"
filetype = ".txt"
f = open("phone_numbers.txt", "w")


for filename in os.listdir(directory):
      if filename.endswith(filetype):
      with open(directory + "/" + filename, 'r') as file:
            if re.search(r, file.read()):
                  f.write(file.name + "\n")

f.close()
```

The problem also has an additional caveat of needing to be done on a short 2-day timeline. While I believe that this script should be capable of doing that as it is, we can improve its speed if necessary by using multithreading and processing the .html files in batches.