// True / False
15. d. File and Scanner
16. b. close the file
17. d. PrintWriter
18. b. Scanner
19. b. resource leak

// algoritmic workbench

17.

```
var writer = new PrintWriter("NumberList.txt);
for (int i = 1; i <= 100; i++)
  writer.println(i);
writer.close();
```

19.

```
var file = new File("NumberList.txt");
var scanner = new Scanner(file);
while (scanner.hasNextLine()){
  System.out.println(scanner.nextLine());
}
scanner.close();
```

21.

```
try (var fileWriter = new FileWriter("NumberList.txt", true);
  var writer = new PrintWriter(fileWriter)){
}
catch (IOException ex){
   ex.print
}
```

// short answers

17. because otherwise its resources will be leaked
19. same as with regular print and println - println will create a new line at the end
21. A immesurable amount of errors can occur. Off the top of my head,
  an attempt to open and rewrite the file contents by another thread/process
  or even a user just accidently removing that file while it is being open

23.

```
FileWriter fw= new FileWriter(path, true);
PrintWriter pw = new PrintWriter(fw);
```