Static Code analysis hands-on:

Instruction:

In all assignments, as part of fixing the reported issues, add comments in the code spacifying change details above code changes and share the final solutions.

Getting Started

1. Login into the Linux server

2. Create a new directory called splint in your home directory <home>

mkdir splint

3. Go inside the directory you have created in (2) /<home>/splint

cd splint

4. Copy the following files from the path as mentioned by the trainer:

a. sample1.c

b. sample2.c

c. sample3.c

d. sample4.c

e. sample5.c

f. sample6.c

Static Code analysis using Splint

```
18:19 ./
18:19 ../
18:00 sample1.c
18:19 sample2.c
18:01 sample3.c
18:01 sample4.c
18:01 sample5.c
18:02 sample6.c
```

5. Read through the code for sample1.c and statically check the file

splint sample1.c

Closely analyze the warnings given by Splint. Some of the warnings given by a static code analyzer may not be valid for your code.

E.g. suppose in this example you do not want the warnings related to unused parameters and variables. Try giving the splint command with –paramuse and –varuse to inhibit these warnings:

splint –paramuse –varuse sample1.c

```
Splint 3.1.2 --- 20 Feb 2018

sample1.c: (in function main)
sample1.c:25:18: Incompatible types for + (char, double): ch + ((double)j - z)
  Types are incompatible. (Use -type to inhibit warning)
sample1.c:25:5: Assignment of double to int: i = (double)(ch + ((double)j - z))
  To allow all numeric types to match, use +relaxtypes.
sample1.c:26:17: Incompatible types for + (long int, char): a + ch
  To make char and int types equivalent, use +charint.
sample1.c:26:5: Assignment of float to int: j = (float)(a + ch + z)
sample1.c:34:5: Unrecognized identifier: foobar
  Identifier used in code has not been declared. (Use -unrecog to inhibit
  warning)
sample1.c:34:5: Unreachable code: foobar(i, j, k)
  This code will never be reached on any possible execution. (Use -unreachable
  to inhibit warning)

Finished checking --- 6 code warnings
```

6. Read through the code for sample2.c and statically check the file

splint sample2.c

```
Splint 3.1.2 --- 20 Feb 2018

sample2.c: (in function main)
sample2.c:13:33: Variable p used before definition
  An rvalue is used that may not be initialized to a value on some execution
  path. (Use -usedef to inhibit warning)

Finished checking --- 1 code warning
```

Edit this file to fix all the warnings and re-run splint on the updated program

```
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
user46@trainux01:~/splinaS
```

7. Read through the code for sample3.c and statically check the file

splint sample3.c

```
Splint 3.1.2 --- 20 Feb 2018

sample3.c: (in function main)
sample3.c:27:21: Possibly null storage data1 passed as non-null param:
                    scanf (..., data1, ...)
  A possibly null pointer is passed as a parameter corresponding to a formal
  parameter with no /*@null@*/ annotation.  If NULL may be used for this
  parameter, add a /*@null@*/ annotation to the function parameter declaration.
  (Use -nullpass to inhibit warning)
   sample3.c:21:17: Storage data1 may become null
sample3.c:27:9: Return value (type int) ignored: scanf("%s", data1)
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
sample3.c:34:9: Index of possibly null pointer data2: data2
  A possibly null pointer is dereferenced.  Value is either the result of a
  function which may return null (in which case, code should check it is not
  null), or a global, parameter or structure field declared with the null
  qualifier. (Use -nullderef to inhibit warning)
   sample3.c:33:17: Storage data2 may become null
sample3.c:44:14: Test expression for while not boolean, type int: 1
  Test expression type is not boolean or int. (Use -predboolint to inhibit
  warning)
sample3.c:51:3: Parse Error. (For help on parse errors, see splint -help
              parseerrors.)
*** Cannot continue.
```

Edit this file to fix all the warnings and re-run splint on the updated program

```
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
```

8. Read through the code for sample4.c and statically check the file

splint sample4.c

```
Splint 3.1.2 --- 20 Feb 2018

sample4.c: (in function main)
sample4.c:37:27: Variable tmp used after being released
  Memory is used after it has been released (either by passing as an only param
  or assigning to an only global). (Use -usereleased to inhibit warning)
   sample4.c:32:11: Storage tmp released
sample4.c:42:14: Fresh storage data2 not released before return
  A memory leak has been detected. Storage allocated locally is not released
  before the last reference to it is lost. (Use -mustfreefresh to inhibit
  warning)
   sample4.c:24:5: Fresh storage data2 created

Finished checking --- 2 code warnings
```

Edit this file to fix all the warnings and re-run splint on the updated program

```
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
```

9. Read through the code for sample5.c and statically check the file

splint sample5.c

```
Splint 3.1.2 --- 20 Feb 2018

sample5.c: (in function func)
sample5.c:31:14: Function exit expects arg 1 to be int gets boolean: true
  To make bool and int types equivalent, use +boolint.
sample5.c:31:14: Argument to exit has implementation defined behavior: true
  The argument to exit should be 0, EXIT_SUCCESS or EXIT_FAILURE (Use -exitarg
  to inhibit warning)
sample5.c:36:12: Return value type int ** does not match declared type int *:
                    &a
  Types are incompatible. (Use -type to inhibit warning)
sample5.c:36:15: Fresh storage a not released before return
  A memory leak has been detected. Storage allocated locally is not released
  before the last reference to it is lost. (Use -mustfreefresh to inhibit
  warning)
   sample5.c:28:37: Fresh storage a created

Finished checking --- 4 code warnings
```

Edit this file to fix all the warnings and re-run splint on the updated program

```
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
```

10. Read through the code for sample6.c and statically check the file

splint sample6.c

```
Splint 3.1.2 --- 20 Feb 2018

sample6.c: (in function checkvalue)
sample6.c:35:2: Path with no return in function declared to return int
  There is a path through a function declared to return a value on which there
  is no return statement. This means the execution may fall through without
  returning a meaningful result to the caller. (Use -noret to inhibit warning)
sample6.c: (in function to_roman)
sample6.c:55:2: Path with no return in function declared to return int
sample6.c:43:7: Variable rom_pos declared but not used
  A variable is declared but never used. Use /*@unused@*/ in front of
  declaration to suppress message. (Use -varuse to inhibit warning)
sample6.c: (in function main)
sample6.c:70:3: Return value (type int) ignored: checkvalue(low)
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
sample6.c:73:6: Return value (type int) ignored: checkvalue(high)
sample6.c:83:20: Passed storage roman not completely defined (*roman is
                    undefined): to_roman (..., roman)
  Storage derivable from a parameter, return value or global is not defined.
  Use /*@out@*/ to denote passed or returned storage which need not be defined.
  (Use -compdef to inhibit warning)
sample6.c:83:5: Return value (type int) ignored: to_roman(low, roman)
sample6.c:91:2: Path with no return in function declared to return int
sample6.c:19:5: Variable exported but not used outside sample6: pows
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
sample6.c:24:7: Variable exported but not used outside sample6: roms
sample6.c:29:5: Function exported but not used outside sample6: checkvalue
   sample6.c:35:1: Definition of checkvalue
sample6.c:37:5: Function exported but not used outside sample6: to_roman
   sample6.c:55:1: Definition of to_roman

Finished checking --- 12 code warnings
```

Edit this file to fix all the warnings and re-run splint on the updated program

```
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
```

Including Static Code analysis as part of the makefile

11. Copy the files below to your working directory (which were used in makefile assignment). Create the project directory structure and copy them to appropriate directory. Add a makefile in make directory to include options to run splint tool on files program.c, simplelink.c. Fix the issues reported.

a. program.c

b. simplelink.h

c. simplelink.c

[You may reuse the makefile created earlier and edit to include splint static analysis]

```
splint -I../include/ ../src/simplelink.c
Splint 3.1.2 --- 20 Feb 2018

../include/simplelink.h:1:6: Variable exported but not used outside simplelink:
                                gsimple
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
../include/simplelink.h:2:6: Variable exported but not used outside simplelink:
                                gcomplex

Finished checking --- 2 code warnings
```