SE LAB 1

SAHIL BONDRE: U18CO021

- Dereferencing a possibly null pointer.
- Using possibly undefined storage or returning storage that is not properly defined.
- Type mismatches, with greater precision and flexibility than provided by C compilers .
- Violations of information hiding.
- Memory management errors including uses of dangling references and memory leaks.
- Dangerous aliasing.

Q1:

```c
#include "stdio.h"

int main() {

  int* d = NULL;

  int x = *d;



  return 0;

}
```

```
→  q-01 git:(master) x gcc main.c
→  q-01 git:(master) x splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function main)
main.c:5:12: Dereference of null pointer d: *d
  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)
   main.c:4:12: Storage d becomes null
main.c:5:7: Variable x 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)

Finished checking --- 2 code warnings
→  q-01 git:(master) x
```

Q2:

```c
extern void foo(/*@out@*/ int *x);

extern int bar(/*@in@*/ int *x);

extern int baz(int *x);

int func(/*@out@*/ int *x, int i) {

  if (i > 3) return *x;

  if (i > 1) return bar(x);

  if (i == 0) return baz(x);

  foo(x);

  return *x;

}

int main() { return 0; }
```

```
→  q-02 git:(master) ✗ gcc -c main.c
→  q-02 git:(master) ✗ splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function func)
main.c:8:21: Value *x used before definition
  An rvalue is used that may not be initialized to a value on some execution
  path. (Use -usedef to inhibit warning)
main.c:9:25: Passed storage x not completely defined (*x is undefined): bar (x)
  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)
main.c:10:26: Passed storage x not completely defined (*x is undefined):
                baz (x)

Finished checking --- 3 code warnings
```

q3:

```c
#include "stdio.h"



int main() {
```

```c
  int x = "45";

  int y = 45.7585f;



  printf("%d %d", x, y);

  return 'c';

}
```

```
→  q-03 git:(master) ✗ gcc -c main.c
main.c: In function ‘main’:
main.c:5:11: warning: initialization of ‘int’ from ‘char *’ makes integer from pointer without a cast [-Wint-conversion]
    5 |   int x = "45";
      |           ^~~~
→  q-03 git:(master) ✗ splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function main)
main.c:5:11: Variable x initialized to type char *, expects int: "45"
  Types are incompatible. (Use -type to inhibit warning)
main.c:6:11: Variable y initialized to type float, expects int: 45.7585f
  To allow all numeric types to match, use +relaxtypes.
main.c:9:10: Return value type char does not match declared type int: 'c'
  A character constant is used as an int. Use +charintliteral to allow
  character constants to be used as ints.  (This is safe since the actual type
  of a char constant is int.)

Finished checking --- 3 code warnings
→  q-03 git:(master) ✗
```

q4:

```c
#include "util.h"

#include "stdio.h"



void foo(abstract_type x) {

  printf("%d ", x);

}



void bar() {

  foo(45);

}
```

```
→  q-04 git:(master) ✗ gcc -c main.c
→  q-04 git:(master) ✗ splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function foo)
main.c:5:17: Format argument 1 to printf (%d) expects int gets abstract_type: x
  Underlying types match, but abstract_type is an abstract type that is not
  accessible here.
   main.c:5:12: Corresponding format code
main.c: (in function bar)
main.c:9:7: Function foo expects arg 1 to be abstract_type gets int: 45
  Underlying types match, but abstract_type is an abstract type that is not
  accessible here.
main.c:4:6: Function exported but not used outside main: foo
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   main.c:6:1: Definition of foo

Finished checking --- 3 code warnings
→  q-04 git:(master) ✗
```

Q5:

```c
#include <stdlib.h>


extern /*@only@*/ int *global;


// only means reference not shared

/*@only@*/ int *foo(/*@only@*/ int *x, int *y, int *z) /*@globals global;@*/ {

  int *m = (int *)malloc(sizeof(int));

  global = y;

  free(x);

  *m = *x;

  return z;

}
```

```
→  q-05 git:(master) ✗ gcc -c main.c
→  q-05 git:(master) ✗ splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function foo)
main.c:10:3: Only storage global (type int *) not released before assignment:
                global = y
  A memory leak has been detected. Only-qualified storage is not released
  before the last reference to it is lost. (Use -mustfreeonly to inhibit
  warning)
   main.c:3:24: Storage global becomes only
main.c:10:3: Implicitly temp storage y assigned to only: global = y
  Temp storage (associated with a formal parameter) is transferred to a
  non-temporary reference. The storage may be released or new aliases created.
  (Use -temptrans to inhibit warning)
main.c:14:4: Dereference of possibly null pointer m: *m
  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)
   main.c:8:12: Storage m may become null
main.c:14:9: Variable x 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)
   main.c:12:8: Storage x released
main.c:16:10: Implicitly temp storage z returned as only: z
main.c:16:12: Fresh storage m 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)
   main.c:8:39: Fresh storage m created

Finished checking --- 6 code warnings
→  q-05 git:(master) ✗
```

Q6:

```c
#include <string.h>

void capitalize(/*@out@*/ char *s, char *t) {

  strcpy(s, t);

}
```

```
→  q-06 git:(master) ✗ gcc -c main.c
→  q-06 git:(master) ✗ splint main.c
Splint 3.1.2 --- 20 Feb 2018

main.c: (in function capitalize)
main.c:4:10: Parameter 1 (s) to function strcpy is declared unique but may be
                aliased externally by parameter 2 (t)
  A unique or only parameter may be aliased by some other parameter or visible
  global. (Use -mayaliasunique to inhibit warning)

Finished checking --- 1 code warning
```