CSL 740: Software Engineering
IIT Jammu, SPRING Semester, 2019-20
Lab Assignment No 1: Static Analysis using Splint

Date uploaded/emailed :5th January 2020

Instructions:

I. *The date of submission: 20th January 2020. This is a hard deadline.*
II. ***Maximum Points 100. Equally distributed.***

1. Analyze and briefly discuss how Rice′s theorem applies to Static Analysis.

2. Run the code in Listing 1 with splint and identify the error. Modify the code to correct it, so that splint does not flag any errors when compiled.

Listing 1: Problem 2

```
int main() {
        char *p;
        int x,
        char* foo();
        printf("Please enter the value of x \n");
        scanf("%d", &x);
        if (x==0)
                p = 0;
        else
           p = foo();
        if (x !=0)
                s=*p;
        else
        printf("The value of x entered is %d\n", x);
        return;
}
```

3. Run the the code in Listing 2 with splint and identify the error. Modify the code to correct it, so that splint does not flag any errors when compiled.

Listing 2: Problem 3

```
#include <stdio.h>
#include <stdlib.h>
#define  MAXTMP 80
void static foo(void)
{
        char *tmp;
        tmp = (char *) malloc(MAXTMP);
        *tmp = 'X   ;
        free(tmp);
}
int main()
{
        foo();
        return 0;
}
```

4. Run the the code in Listing 3 with splint and identify the error. Modify the code to correct it, so that splint does not flag any errors when compiled.

Listing 3: Problem 4

```
int main() {
int *p = NULL;   // line number 2
int test;
(void) scanf(”%d”, &test);
if(test > 0)
        p = &test;
else
        *p = 123;        // line number 8
        return 1;
}
```

5. Run the the code in Listing 4 with splint and identify the error. Modify the code to correct it.

Listing 4: Problem 5

```
#include <stdlib.h>
struct check {
        char *sname;
        size_t ncount;
};
static int f1(struct check *testc) {
        char \*b = (char \*)malloc(sizeof(char));
        if(b == NULL)    return 0;
        printf(”Input String: ”);
        (void) scanf(”%s”, b);
        testc->sname = b;
        testc->ncount = strlen(b);
        return 1;
}
static char* f2(){
        char *str =
        (char*) malloc(sizeof(char));
        if(str != NULL)
                strcpy(str, ”TESTING”);
        return str;
}
int main(){
        struct check *c =
        (struct check*)malloc(sizeof(struct check));

        if(c==NULL)
                exit(0);
        if(f1(c) == 0) {
                if(c->sname != NULL)
                        free(c->sname);
                c->sname = f2();
                if(c->sname != NULL)
                        c->ncount = strlen(c->sname);
        }
        if(c != NULL)
                free(c);
        return(1);
}
```

6. Use the /@*null*@/ and the /@*in*@/, /@*out*@/ annotations to ensure correctness in execution in Problem #5 in this assignment.

7. Run splint on the code shown in Listing 5 and explain the errors:.

Listing 5: Problem 7

```
#include <stdlib.h>
static int* f1() {
        int value;
        printf("Input Number: ");
        (void) scanf("%d", &value);
        return \&value;
}
static char* f2() {
        return "TESTING";
}
int main() {
        int *retvalue;
        char *str = (char *)malloc(sizeof(char));
        retvalue = f1();
        if(*retvalue > 0 && str != NULL)  {
                strcpy(str, f2());
                printf("String: %s \n", str);
        }
        if(str != NULL)
                free(str);
        if(retvalue != NULL)
                free(retvalue);
        return(1);
}
```

8. Using the /@*null*@/, the /@*observer*@/, /@*only*@/ annotations, correct the program in Problem #7, here.

9. Run the the code in Listing 6 with splint and identify the errors, if any. Modify the code with the splint buffer overflow check invariant maxSet(), appropriately. Comment your program code to explain how splint models blocks of contiguous memory using two properties:maxSetandmaxRead, as also how are the invariants inserted by splint as pre and post conditions to check of anomalous code.

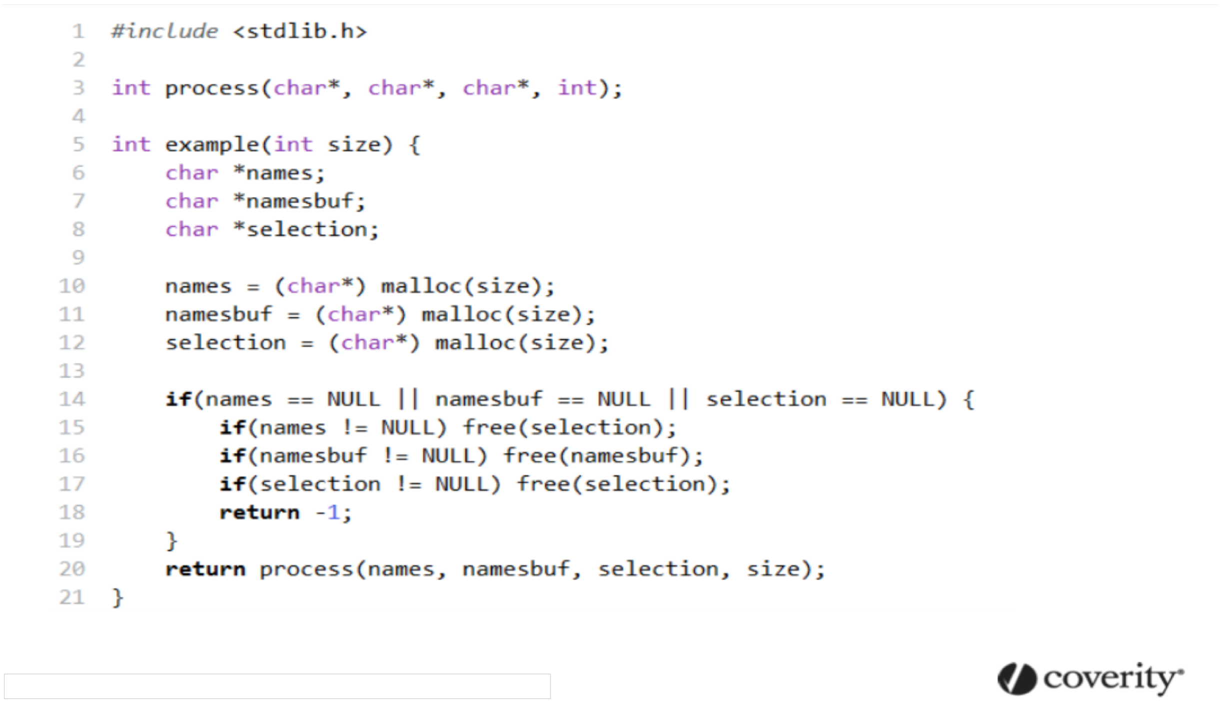
Listing 6: Problem 9

```
/*include the files <stdio.h> <string.h> <stdlib.h>. <stddef.h>
void static updateEnv(char *str, size_t size)
{
        char *tmp;
        tmp = getenv("HOME");
        if (tmp != NULL) {
                strncpy(str,tmp,size -1);
                str[size -1] = '\0    ;
        }
}
int main() {
        char *str;
        size_t size;
        str = "Hello World";
        size = strlen(str);
        updateEnv(str, (size -1));
        printf("\nThe Environment variable copied\n");
        return 0;
}
```

10. Use the program uploaded viz. Problem10Assign1.c and explain all the errors flagged of by splint.

11. Use the program uploaded viz. Problem11Assign1.c and explain all the errors flagged of by splint.

12. Consider the code shown in Figure 1 here. First logically analyze the code and comment whether it contains any errors or not. Insert appropriate main() function for the code to compile correctly with gcc. Then, compile the modified code splint and comment on the errors flagged. Use your observations to answer the following question: *Can the lifetime of a variable exceed its scope ?* Justify your answer with appropriate illustration(s).



```c
#include <stdlib.h>

int process(char*, char*, char*, int);

int example(int size) {
    char *names;
    char *namesbuf;
    char *selection;

    names = (char*) malloc(size);
    namesbuf = (char*) malloc(size);
    selection = (char*) malloc(size);

    if(names == NULL || namesbuf == NULL || selection == NULL) {
        if(names != NULL) free(selection);
        if(namesbuf != NULL) free(namesbuf);
        if(selection != NULL) free(selection);
        return -1;
    }
    return process(names, namesbuf, selection, size);
}
```

Figure 1: Code for Problem 12