

Assignment 3

Submitted by U17CO110

Question 1:

i) Infinite Loops

Program

```
Q1_1.c > ...
1  #include <stdio.h>
2
3  extern int glob1, glob2;
4  extern int f(void) /*@globals glob1@/ /*@modifies nothing@/;
5  extern void g(void) /*@modifies glob2@/;
6  extern void h(void);
7
8  void upto(int x)
9  {
10     while (x > f())
11         g();
12     while (f() < 3)
13         h();
14 }
15
16 int main()
17 {
18     printf("Checking for likely infinite loops\n");
19     return 0;
20 }
```

GCC Output

```
C:\WINDOWS\system32\cmd.exe

C:\splint-3.1.2\lib>gcc A3_Q1.1.c

C:\splint-3.1.2\lib>a
Checking for likely infinite loops

C:\splint-3.1.2\lib>_
```

Splint Output

```
C:\splint-3.1.2\lib>splint A3_Q1.1.c -preproc +infloopsuncon
Splint 3.1.2 --- 25 Aug 2010

A3_Q1.1.c: (in function upto)
A3_Q1.1.c(9,12): Suspected infinite loop. No value used in loop test (x,
glob1) is modified by test or loop body.
    This appears to be an infinite loop. Nothing in the body of the loop or the
    loop test modifies the value of the loop test. Perhaps the specification of a
    function called in the loop body is missing a modification. (Use -infloops to
    inhibit warning)
A3_Q1.1.c(10,12): Suspected infinite loop. No condition values modified.
    Modification possible through unconstrained calls: h
    This appears to be an infinite loop. Nothing in the body of the loop or the
    loop test modifies the value of the loop test. There may be a modification
    through a call to an unconstrained function, or an unconstrained function in
    the loop test may use a global variable modified by the loop body. (Use
    -infloopsuncon to inhibit warning)

Finished checking --- 2 code warnings

C:\splint-3.1.2\lib>
```

ii) Fall through switch cases

Program

```
#include <stdio.h>
typedef enum { YES, NO, DEFINITELY, PROBABLY, MAYBE } ynm;

void decide(ynm y)
{
    switch (y)
    {
        case PROBABLY:
        case NO:
            printf("No!");
        case MAYBE:
            printf("Maybe");
        /*@fallthrough@*/
        case YES:
            printf("Yes!");
        case DEFINITELY:
            printf("Definitely!");
    }
}

int main()
{
    printf("Checking fall through switch cases");
    return 0;
}
```

GCC Output

```
C:\WINDOWS\system32\cmd.exe

C:\splint-3.1.2\lib>gcc A3_Q1.2.c

C:\splint-3.1.2\lib>a
Checking fall through switch cases
C:\splint-3.1.2\lib>
```

Splint Output

```
C:\splint-3.1.2\lib>splint A3_Q1.2.c -preproc
Splint 3.1.2 --- 25 Aug 2010

A3_Q1.2.c: (in function decide)
A3_Q1.2.c(10,18): Fall through case (no preceding break)
    Execution falls through from the previous case (use /*@fallthrough@*/ to mark
    fallthrough cases). (Use -casebreak to inhibit warning)
A3_Q1.2.c(13,18): Fall through case (no preceding break)

Finished checking --- 2 code warnings

C:\splint-3.1.2\lib>
```

iii) Missing switch case

Program

```
#include <stdio.h>

typedef enum { YES, NO, DEFINITELY, PROBABLY, MAYBE } ynm;

void decide(ynm y)
{
    switch (y)
    {
        case PROBABLY:
            break;
        case NO:
            printf("No!");
            break;
        case MAYBE:
```

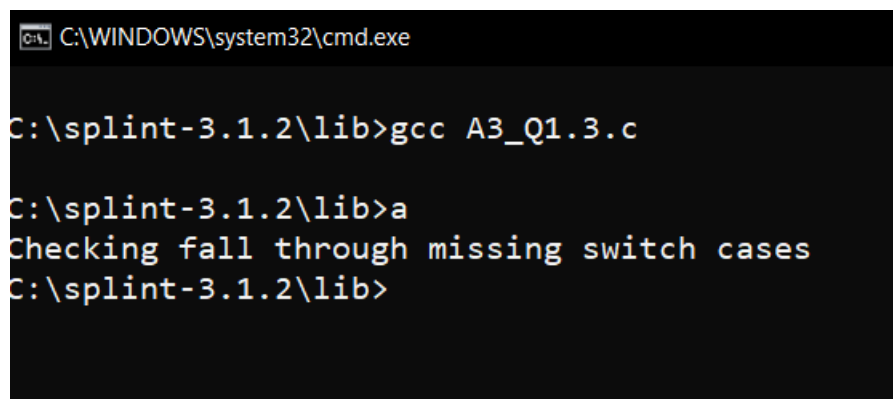
```

        printf("Maybe");
        break;
    /*@fallthrough@*/
    case YES:
        printf("Yes!");
        break;
    }
}

int main()
{
    printf("Checking fall through missing switch cases");
    return 0;
}

```

GCC Output



```

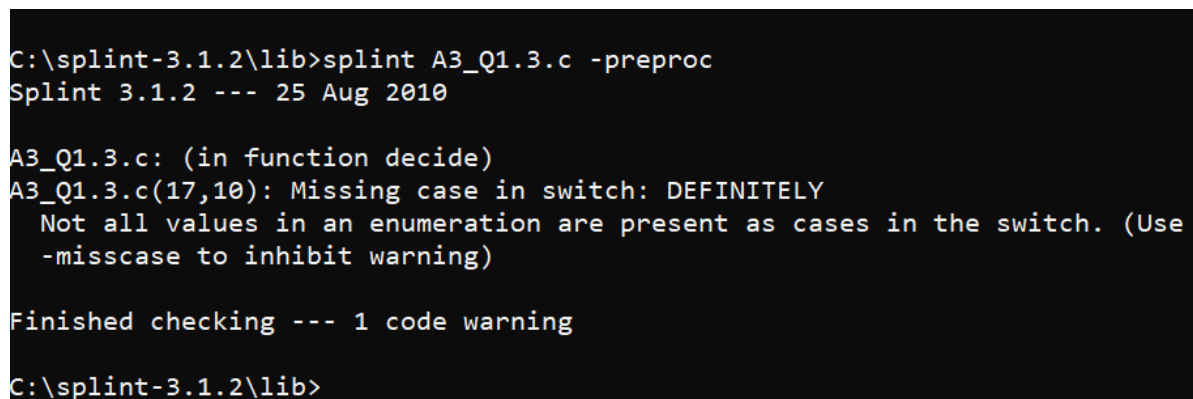
C:\WINDOWS\system32\cmd.exe

C:\splint-3.1.2\lib>gcc A3_Q1.3.c

C:\splint-3.1.2\lib>a
Checking fall through missing switch cases
C:\splint-3.1.2\lib>

```

Splint Output



```

C:\splint-3.1.2\lib>splint A3_Q1.3.c -preproc
Splint 3.1.2 --- 25 Aug 2010

A3_Q1.3.c: (in function decide)
A3_Q1.3.c(17,10): Missing case in switch: DEFINITELY
    Not all values in an enumeration are present as cases in the switch. (Use
    -misscase to inhibit warning)

Finished checking --- 1 code warning

C:\splint-3.1.2\lib>

```

- iv) Empty statement after an if, while or for statement

Program

```
#include <stdio.h>

int main()
{
    int x = 1;
    if (x > 3)
        ;
    if (x > 3)
        x++;

    printf("Checking Empty statement after an if , while or for\n");
    return 0;
}
```

GCC Output

```
C:\WINDOWS\system32\cmd.exe

C:\splint-3.1.2\lib>gcc A3_Q1.3.c

C:\splint-3.1.2\lib>a
Checking fall through missing switch cases
C:\splint-3.1.2\lib>
```

Splint Output

```
C:\splint-3.1.2\lib>splint A3_Q1.4.c -preproc
Splint 3.1.2 --- 25 Aug 2010

A3_Q1.4.c: (in function main)
A3_Q1.4.c(6,15): Body of if statement is empty
    If statement has no body. (Use -ifempty to inhibit warning)

Finished checking --- 1 code warning

C:\splint-3.1.2\lib>
```

Question 2:

Buffers are memory storage regions that temporarily hold data while it is being transferred from one location to another. A buffer overflow (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. As a result, the program attempting to write the data to the buffer overwrites adjacent memory locations

Program

```
#include <stdio.h>

void updateEnv(char *str)
{
    char *tmp;
    tmp = getenv("MYENV");
    if (tmp != NULL)
        strcpy(str, tmp);
}

void updateEnvSafe(char *str, size_t strSize) /*@requires maxS
et(str @ A3_Q2.c(18,9)) >= strSize @ A3_Q2.c(18,13)@*/
{
    char *tmp;
    tmp = getenv("MYENV");
    if (tmp != NULL)
    {
        strncpy(str, tmp,
                strSize - 1);
        str[strSize - 1] = '/0';
    }
}

int main()
{
    printf("Buffer Owerflow attack\n");
    return 0;
}
```

The above program shows two ways of updating the environment variables. The first function doesn't update it safely and can lead to buffer overflow attack

whereas the second function is the correct method to update the environment variable and doesn't cause any buffer overflow attack.

GCC Output

```
C:\splint-3.1.2\lib>gcc A3_Q2.c --no-warning

C:\splint-3.1.2\lib>a
Buffer Owerflow attack

C:\splint-3.1.2\lib>
```

Splint Output

```
C:\splint-3.1.2\lib>splint A3_Q2.c -preproc +bounds
Splint 3.1.2 --- 25 Aug 2010

A3_Q2.c: (in function updateEnv)
A3_Q2.c(7,9): Possible out-of-bounds store: strcpy(str, tmp)
  Unable to resolve constraint:
    requires maxSet(str @ A3_Q2.c(7,17)) >= maxRead(getenv("MYENV") @
    A3_Q2.c(5,11))
  needed to satisfy precondition:
    requires maxSet(str @ A3_Q2.c(7,17)) >= maxRead(tmp @ A3_Q2.c(7,22))
    derived from strcpy precondition: requires maxSet(<parameter 1>) >=
    maxRead(<parameter 2>)
  A memory write may write to an address beyond the allocated buffer. (Use
  -boundswrite to inhibit warning)
A3_Q2.c(10,77): Parse Error. (For help on parse errors, see splint -help
  parseerrors.)
*** Cannot continue.

C:\splint-3.1.2\lib>
```

Question 3:

Program

```
#include <stdio.h>

extern int square(/*@sef@*/ int x);
#define square(x) ((x) * (x))
```

```

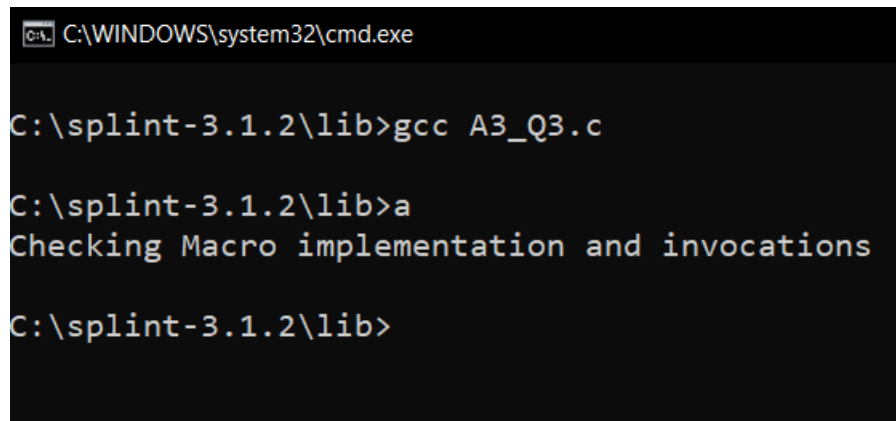
extern int sumsquares(int x, int y);
#define sumsquares(x, y) (square(x) + square(y))

int main()
{
    int i = 1;
    i = square(i++);
    i = sumsquares(i, i);

    printf("Checking Macro implementation and invocations\n");
    return 0;
}

```

GCC Output



```

C:\WINDOWS\system32\cmd.exe

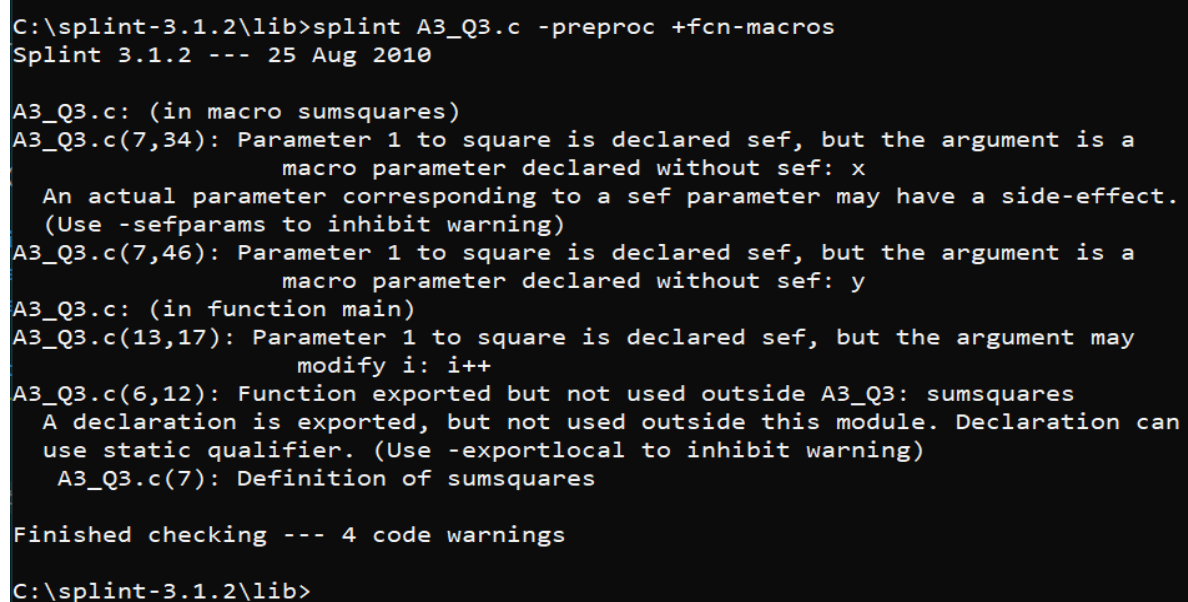
C:\splint-3.1.2\lib>gcc A3_Q3.c

C:\splint-3.1.2\lib>a
Checking Macro implementation and invocations

C:\splint-3.1.2\lib>

```

Splint Output



```

C:\splint-3.1.2\lib>splint A3_Q3.c -preproc +fcn-macros
Splint 3.1.2 --- 25 Aug 2010

A3_Q3.c: (in macro sumsquares)
A3_Q3.c(7,34): Parameter 1 to square is declared sef, but the argument is a
        macro parameter declared without sef: x
    An actual parameter corresponding to a sef parameter may have a side-effect.
    (Use -sefparams to inhibit warning)
A3_Q3.c(7,46): Parameter 1 to square is declared sef, but the argument is a
        macro parameter declared without sef: y
A3_Q3.c: (in function main)
A3_Q3.c(13,17): Parameter 1 to square is declared sef, but the argument may
        modify i: i++
A3_Q3.c(6,12): Function exported but not used outside A3_Q3: sumsquares
    A declaration is exported, but not used outside this module. Declaration can
    use static qualifier. (Use -exportlocal to inhibit warning)
    A3_Q3.c(7): Definition of sumsquares

Finished checking --- 4 code warnings

C:\splint-3.1.2\lib>

```


Question 4:

Program 1:

```
.c / ...  
//modification  
void setx(int *x, int *y) /*@modifies *x@*/  
{  
    *y = *x;  
}  
void sety(int *x, int *y) /*@modifies *y@*/  
{  
    setx(y, x);  
}
```

Splint Output

```
C:\splint-3.1.2\lib>splint ques_3_4_1.c +checks -preproc -incondefs  
Splint 3.1.2 --- 25 Aug 2010  
  
ques_3_4_1.c: (in function setx)  
ques_3_4_1.c(6,1): Undocumented modification of *y: *y = *x  
    An externally-visible object is modified by a function, but not listed in its  
    modifies clause. (Use -mods to inhibit warning)  
ques_3_4_1.c(7,1): Suspect object listed in modifies of setx not modified: *x  
    An object listed in the modifies clause is not modified by the implementation  
    of the function. The modification may not be detected if it is done through a  
    call to an unspecified function. (Use -mustmod to inhibit warning)  
    ques_3_4_1.c(3,6): Declaration of setx  
ques_3_4_1.c(3,6): Function exported but not used outside ques_3_4_1: setx  
    A declaration is exported, but not used outside this module. Declaration can  
    use static qualifier. (Use -exportlocal to inhibit warning)  
    ques_3_4_1.c(7,1): Definition of setx  
ques_3_4_1.c(3,6): Function setx exported but not declared in header file  
    A declaration is exported, but does not appear in a header file. (Use  
    -exportheader to inhibit warning)  
    ques_3_4_1.c(7,1): Definition of setx  
ques_3_4_1.c(8,6): Function sety exported but not declared in header file  
    ques_3_4_1.c(12,1): Definition of sety  
  
Finished checking --- 5 code warnings  
  
C:\splint-3.1.2\lib>
```

Program 2:

```
//global variables
int glob1, glob2;
int f(void) /*@globals glob1;@*/
{
    return glob2;
}
```

Splint Output

```
C:\splint-3.1.2\lib>splint ques_3_4_2.c +checks -preproc -incondefs
Splint 3.1.2 --- 25 Aug 2010

ques_3_4_2.c: (in function f)
ques_3_4_2.c(6,8): Undocumented use of global glob2
  A checked global variable is used in the function, but not listed in its
  globals clause. By default, only globals specified in .lcl files are checked.
  To check all globals, use +allglobals. To check globals selectively use
  /*@checked@*/ in the global declaration. (Use -globals to inhibit warning)
ques_3_4_2.c(4,5): Global glob1 listed but not used
  A global variable listed in the function's globals list is not used in the
  body of the function. (Use -globuse to inhibit warning)
ques_3_4_2.c(3,12): Variable exported but not used outside ques_3_4_2: glob2
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
ques_3_4_2.c(3,5): Variable glob1 exported but not declared in header file
  A variable declaration is exported, but does not appear in a header file.
  (Used with exportheader.) (Use -exportheadervar to inhibit warning)
ques_3_4_2.c(3,12): Variable glob2 exported but not declared in header file
ques_3_4_2.c(4,5): Function f exported but not declared in header file
  A declaration is exported, but does not appear in a header file. (Use
  -exportheader to inhibit warning)
  ques_3_4_2.c(7,1): Definition of f

Finished checking --- 6 code warnings

C:\splint-3.1.2\lib>
```