SE ASSIGNMENT 1

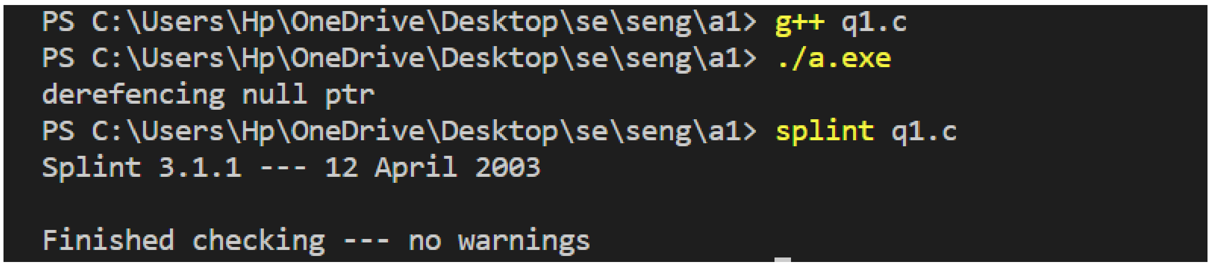
SOURABH PATEL

U19CS082

From the classes of errors listed below, Design a c program fragment to compare the outputs of Splint and the Standard C compiler.

1) Dereferencing a possibly null pointer.

```c
#include <stdio.h>
char firstChar1 ( char *s) {
    return *s;
}
char firstChar2 ( char *s) {
    if (s == NULL)
    return '\0';
    return *s;
}
int main() {
    printf("derefencing null ptr");
    return 0;
}
```

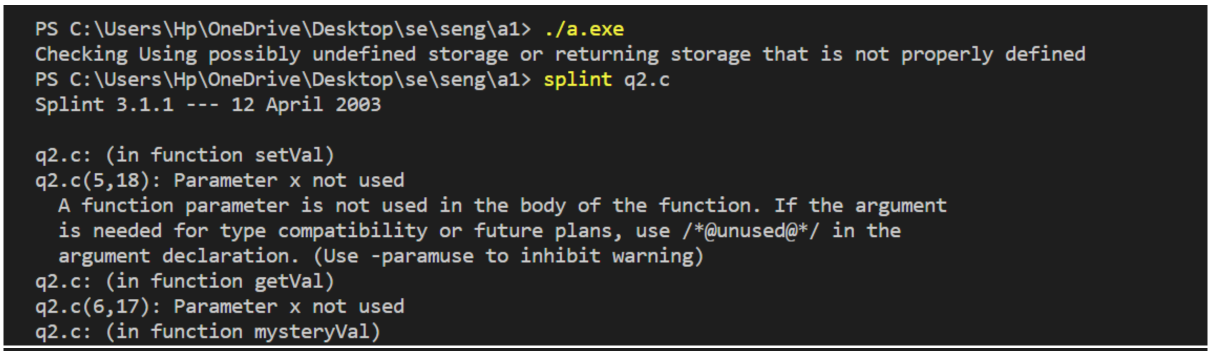


```
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> g++ q1.c
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> ./a.exe
derefencing null ptr
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> splint q1.c
Splint 3.1.1 --- 12 April 2003

Finished checking --- no warnings
```

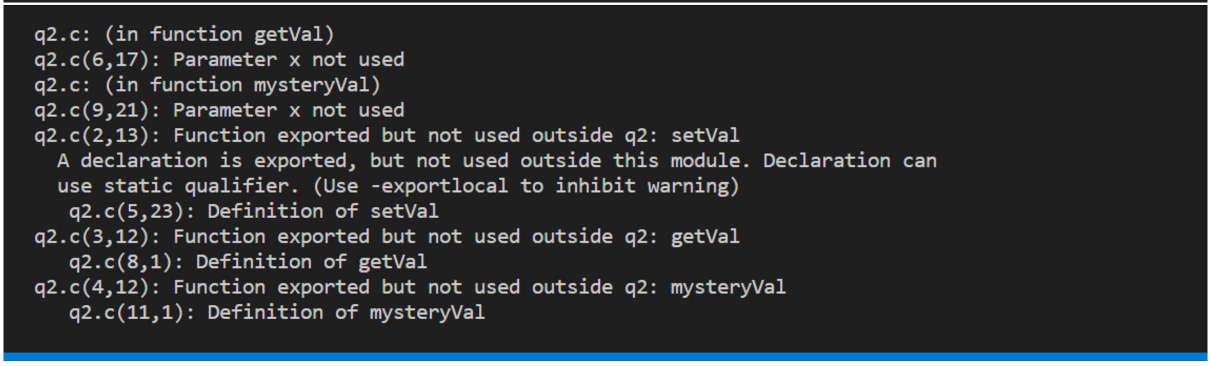
2) Using possibly undefined storage or returning storage that is not properly defined.

```c
#include <stdio.h>
extern void setVal(int *x);
extern int getVal(int *x);
extern int mysteryVal(int *x);
void setVal(int *x) {}
int getVal(int *x){
return 1;
}
int mysteryVal(int *x){
return 1;
}
int dumbfunc(int *x, int i){
```

```c
if (i > 3)
return *x;
else if (i > 1)
return getVal(x);
else if (i == 0)
return mysteryVal(x);
else
{
setVal(x);
return *x;
}
}
int main(){
printf("Checking Using possibly undefined storage or returning storage that is not properly defined");
return 0;
}
```



```
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> ./a.exe
Checking Using possibly undefined storage or returning storage that is not properly defined
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> splint q2.c
Splint 3.1.1 --- 12 April 2003

q2.c: (in function setVal)
q2.c(5,18): Parameter x not used
  A function parameter is not used in the body of the function. If the argument
  is needed for type compatibility or future plans, use /*@unused@*/ in the
  argument declaration. (Use -paramuse to inhibit warning)
q2.c: (in function getVal)
q2.c(6,17): Parameter x not used
q2.c: (in function mysteryVal)
```

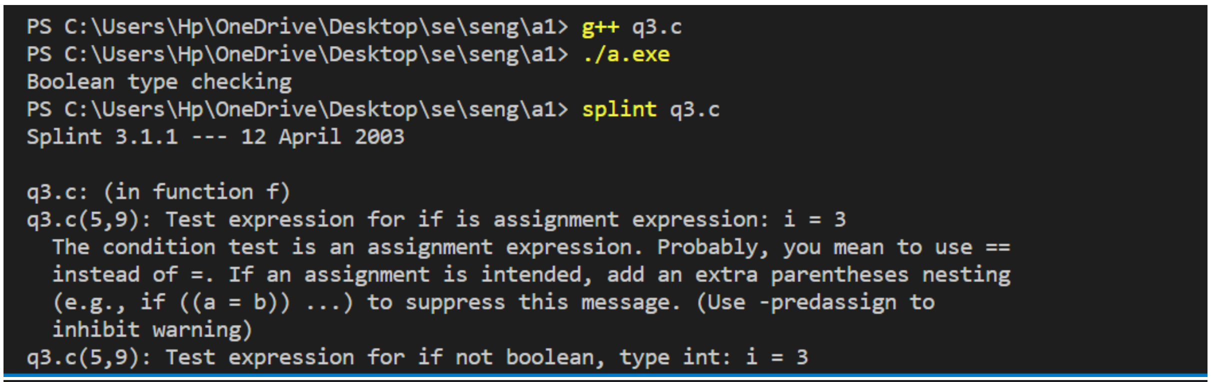


```
q2.c: (in function getVal)
q2.c(6,17): Parameter x not used
q2.c: (in function mysteryVal)
q2.c(9,21): Parameter x not used
q2.c(2,13): Function exported but not used outside q2: setVal
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   q2.c(5,23): Definition of setVal
q2.c(3,12): Function exported but not used outside q2: getVal
   q2.c(8,1): Definition of getVal
q2.c(4,12): Function exported but not used outside q2: mysteryVal
   q2.c(11,1): Definition of mysteryVal
```

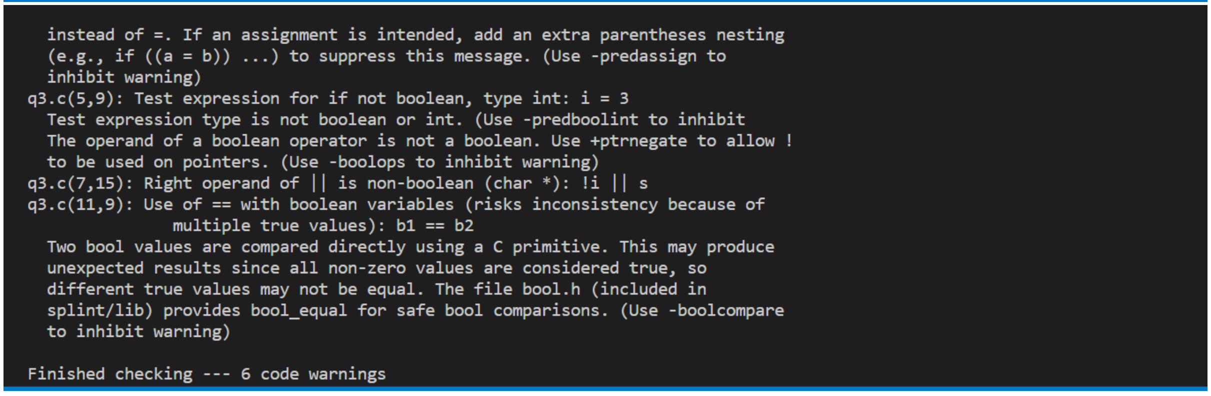
3) Type mismatches, with greater precision and flexibility than provided by C compilers .

```c
#include <stdio.h>
#include <stdbool.h>
```

```c
int f (int i, char *s, bool b1, bool b2) {
    if (i = 3)
        return b1;
    if (!i || s)
        return i;
    if (s)
        return 7;
    if (b1 == b2)
        return 3;

    return 2;
}

  int main() {
    printf("Boolean type checking");

    return 0;
  }
```



```
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> g++ q3.c
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> ./a.exe
Boolean type checking
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> splint q3.c
Splint 3.1.1 --- 12 April 2003

q3.c: (in function f)
q3.c(5,9): Test expression for if is assignment expression: i = 3
  The condition test is an assignment expression. Probably, you mean to use ==
  instead of =. If an assignment is intended, add an extra parentheses nesting
  (e.g., if ((a = b)) ...) to suppress this message. (Use -predassign to
  inhibit warning)
q3.c(5,9): Test expression for if not boolean, type int: i = 3
```



```
  instead of =. If an assignment is intended, add an extra parentheses nesting
  (e.g., if ((a = b)) ...) to suppress this message. (Use -predassign to
  inhibit warning)
q3.c(5,9): Test expression for if not boolean, type int: i = 3
  Test expression type is not boolean or int. (Use -predboolint to inhibit
  The operand of a boolean operator is not a boolean. Use +ptrnegate to allow !
  to be used on pointers. (Use -boolops to inhibit warning)
q3.c(7,15): Right operand of || is non-boolean (char *): !i || s
q3.c(11,9): Use of == with boolean variables (risks inconsistency because of
               multiple true values): b1 == b2
  Two bool values are compared directly using a C primitive. This may produce
  unexpected results since all non-zero values are considered true, so
  different true values may not be equal. The file bool.h (included in
  splint/lib) provides bool_equal for safe bool comparisons. (Use -boolcompare
  to inhibit warning)

Finished checking --- 6 code warnings
```

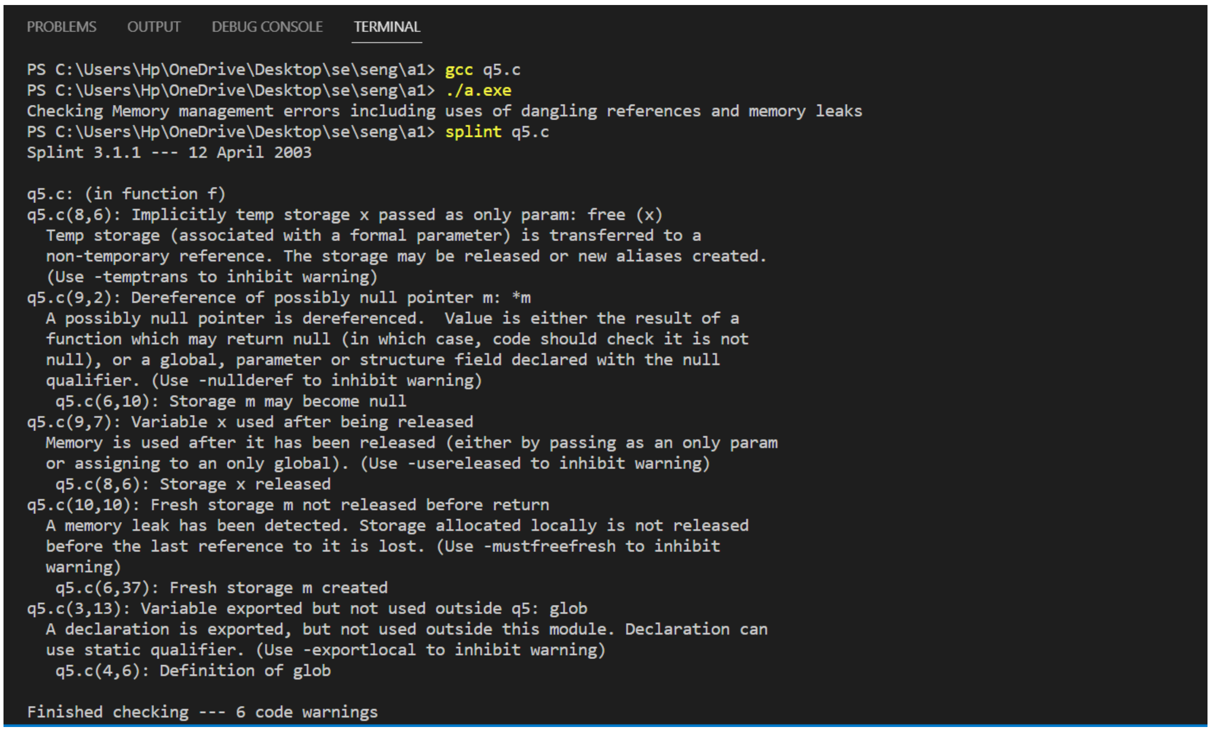
4) Violations of information hiding.

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
```

```
#include <mstring.h>

bool isPalindrome (mstring s) {
    char *current = (char *)s;
    int i, len = (int)strlen (s);
    for (i = 0; i <= (len+1) / 2; i++) {
        if (current[i] != s[len-i-1])
            return false;
    }
  return true;
}

bool callPal (void) {
    return (isPalindrome ("bob"));
}

int main() {
    printf("Information hiding violations");
    return 0;
}
```

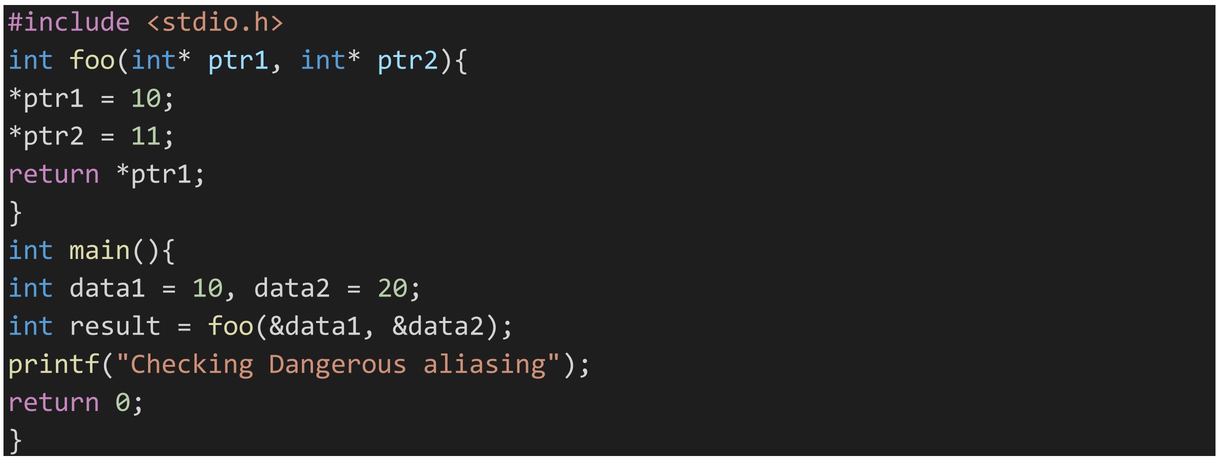
5) Memory management errors including uses of dangling references and memory leaks.

```
#include <stdio.h>
#include <stdlib.h>
extern int *glob;
int *glob;
int *f(int *x, int *y, int *z){
int *m = (int *)malloc(sizeof(int));
glob = y; //Memory leak
free(x);
*m = *x; //Use after free
return z; //Memory leak detected
}
int main(){
printf("Checking Memory management errors including uses of dangling
references and memory leaks");
return 0;
}
```

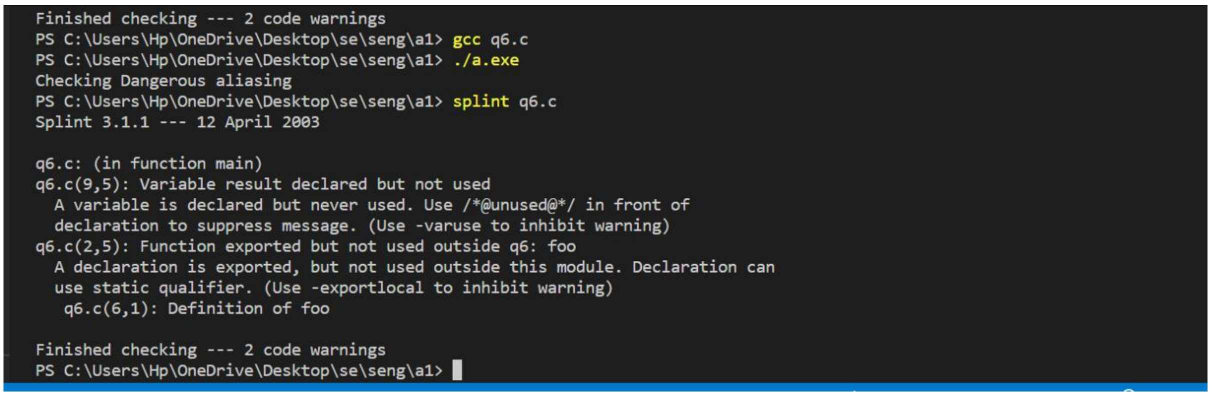

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> gcc q5.c
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> ./a.exe
Checking Memory management errors including uses of dangling references and memory leaks
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> splint q5.c
Splint 3.1.1 --- 12 April 2003

q5.c: (in function f)
q5.c(8,6): Implicitly temp storage x passed as only param: free (x)
  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)
q5.c(9,2): 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)
   q5.c(6,10): Storage m may become null
q5.c(9,7): 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)
   q5.c(8,6): Storage x released
q5.c(10,10): 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)
   q5.c(6,37): Fresh storage m created
q5.c(3,13): Variable exported but not used outside q5: glob
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   q5.c(4,6): Definition of glob

Finished checking --- 6 code warnings
```

6) Dangerous aliasing.



```c
#include <stdio.h>
int foo(int* ptr1, int* ptr2){
*ptr1 = 10;
*ptr2 = 11;
return *ptr1;
}
int main(){
int data1 = 10, data2 = 20;
int result = foo(&data1, &data2);
printf("Checking Dangerous aliasing");
return 0;
}
```



```
Finished checking --- 2 code warnings
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> gcc q6.c
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> ./a.exe
Checking Dangerous aliasing
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> splint q6.c
Splint 3.1.1 --- 12 April 2003

q6.c: (in function main)
q6.c(9,5): Variable result 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)
q6.c(2,5): Function exported but not used outside q6: foo
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   q6.c(6,1): Definition of foo

Finished checking --- 2 code warnings
PS C:\Users\Hp\OneDrive\Desktop\se\seng\a1> 
```