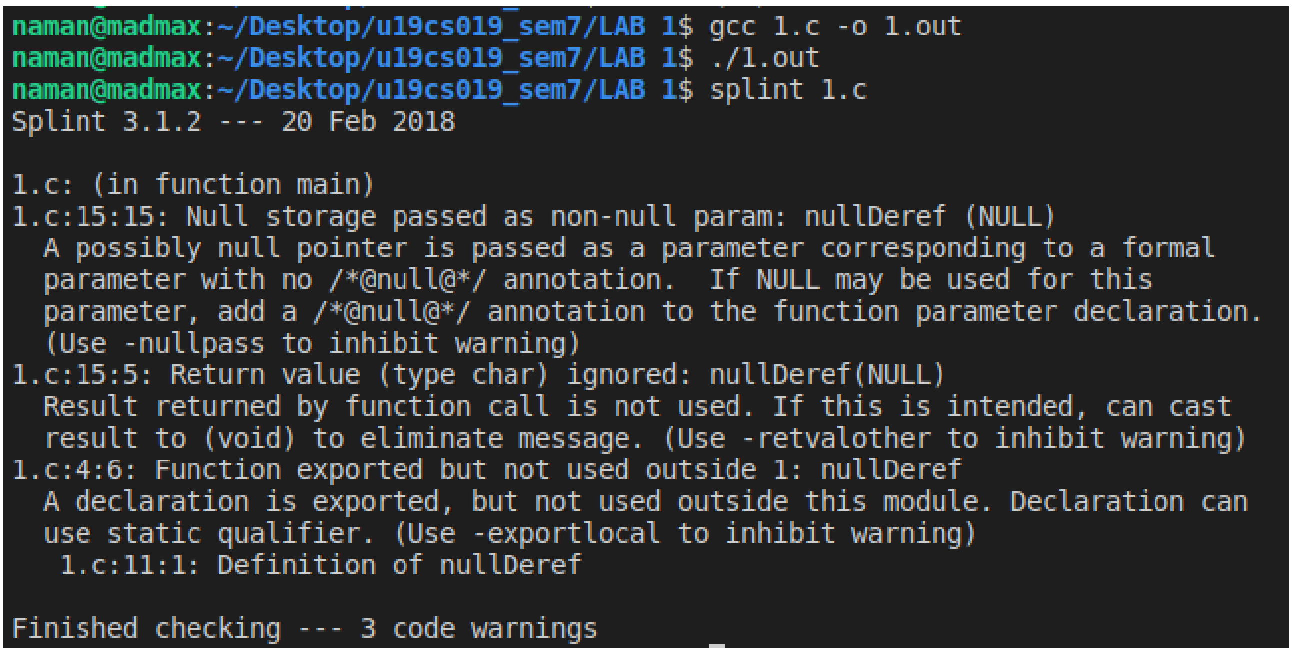
Name: NAMAN KHATER LAB 1 Admission no: U19CS019

1. Dereferencing a possibly null pointer.

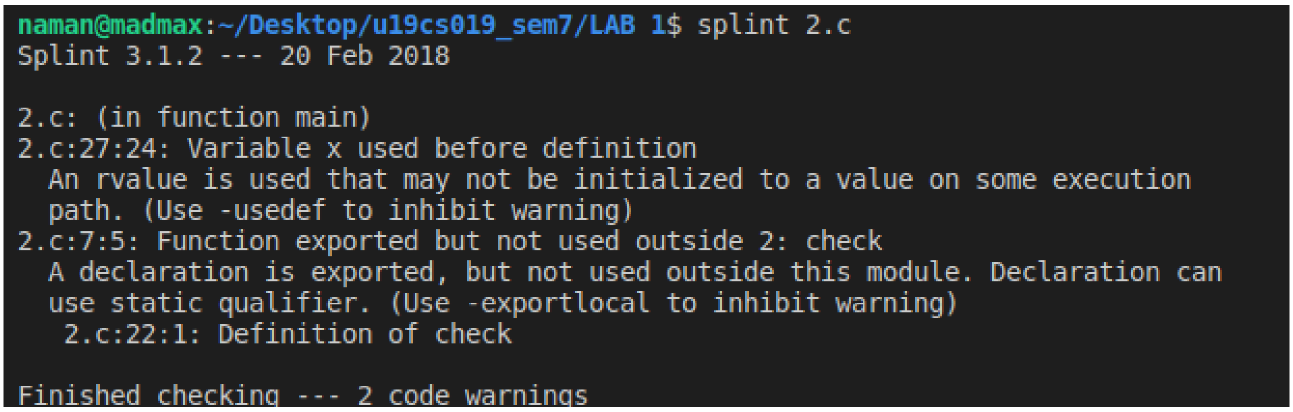
```
#include <stdio.h>
#include <stdbool.h>

char nullDeref(char *s)
{
if (s == NULL)
{
return '\0';
}
return *s;
}

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



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ gcc 1.c -o 1.out
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ ./1.out
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 1.c
Splint 3.1.2 --- 20 Feb 2018

1.c: (in function main)
1.c:15:15: Null storage passed as non-null param: nullDeref (NULL)
  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)
1.c:15:5: Return value (type char) ignored: nullDeref(NULL)
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalother to inhibit warning)
1.c:4:6: Function exported but not used outside 1: nullDeref
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   1.c:11:1: Definition of nullDeref

Finished checking --- 3 code warnings
```

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

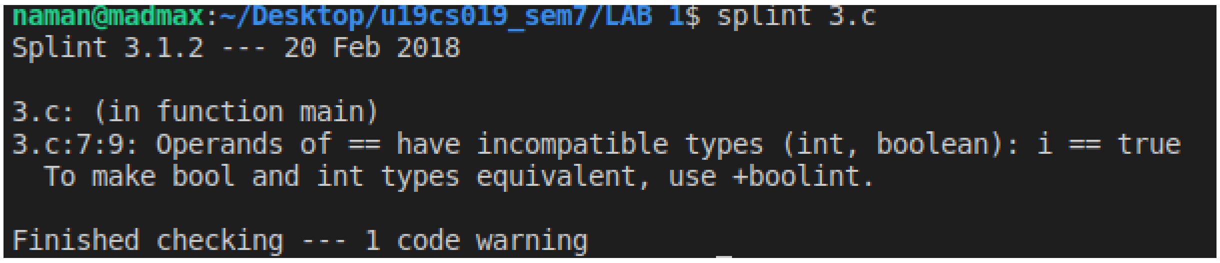
```
#include <stdio.h>
#include <stdbool.h>

extern void setVal(int *x);
extern int getVal(int *x);

int check(int *x, int ch)
{
   if (ch == 1)
   {
       return *x;
   }
   else if (ch == 2)
   {
       return getVal(x);
   }
   else
   {
       setVal(x);
       return *x;
   }
}

int main()
{
   int *x, ch = 4;
   printf("%d", check(x, ch));
   return 0;
}
```



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 2.c
Splint 3.1.2 --- 20 Feb 2018

2.c: (in function main)
2.c:27:24: Variable 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)
2.c:7:5: Function exported but not used outside 2: check
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   2.c:22:1: Definition of check

Finished checking --- 2 code warnings
```

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

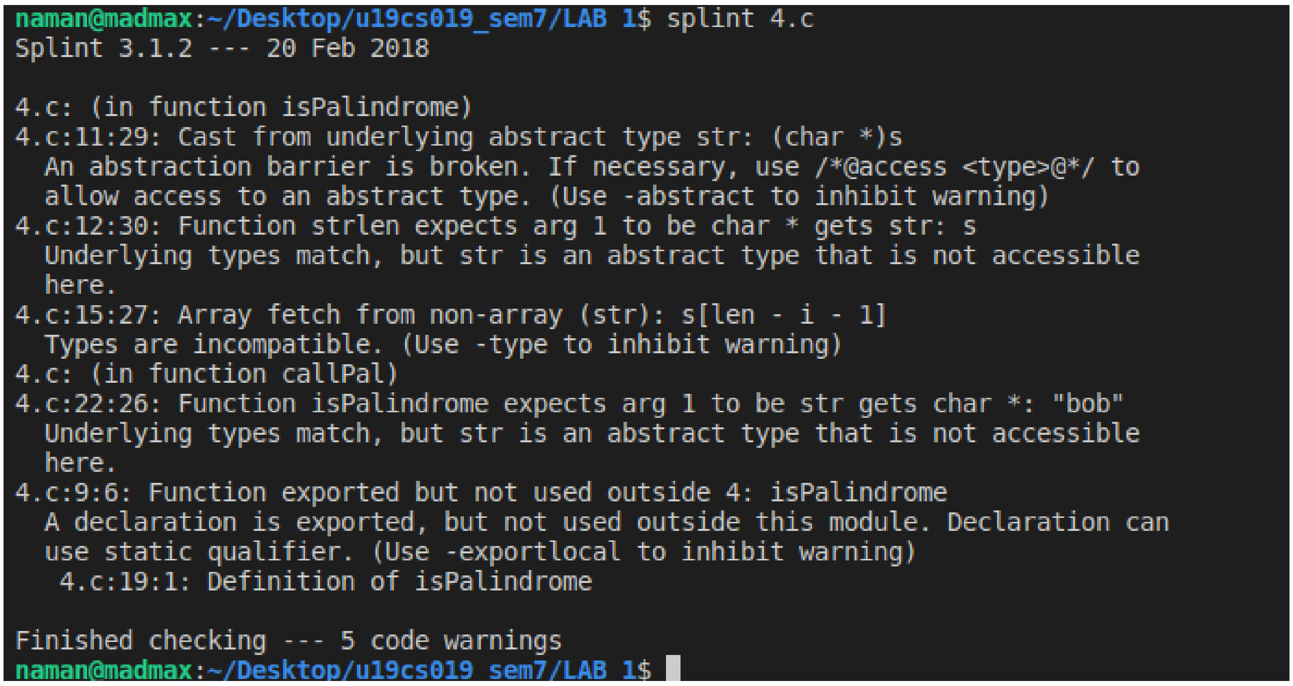
```
#include <stdio.h>
#include <stdbool.h>

int main()
{
    int i = 0;
    if (i == true)
    {
        printf("hi");
    }
    return 0;
}
```



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 3.c
Splint 3.1.2 --- 20 Feb 2018

3.c: (in function main)
3.c:7:9: Operands of == have incompatible types (int, boolean): i == true
  To make bool and int types equivalent, use +boolint.

Finished checking --- 1 code warning
```

4. Violations of information hiding.

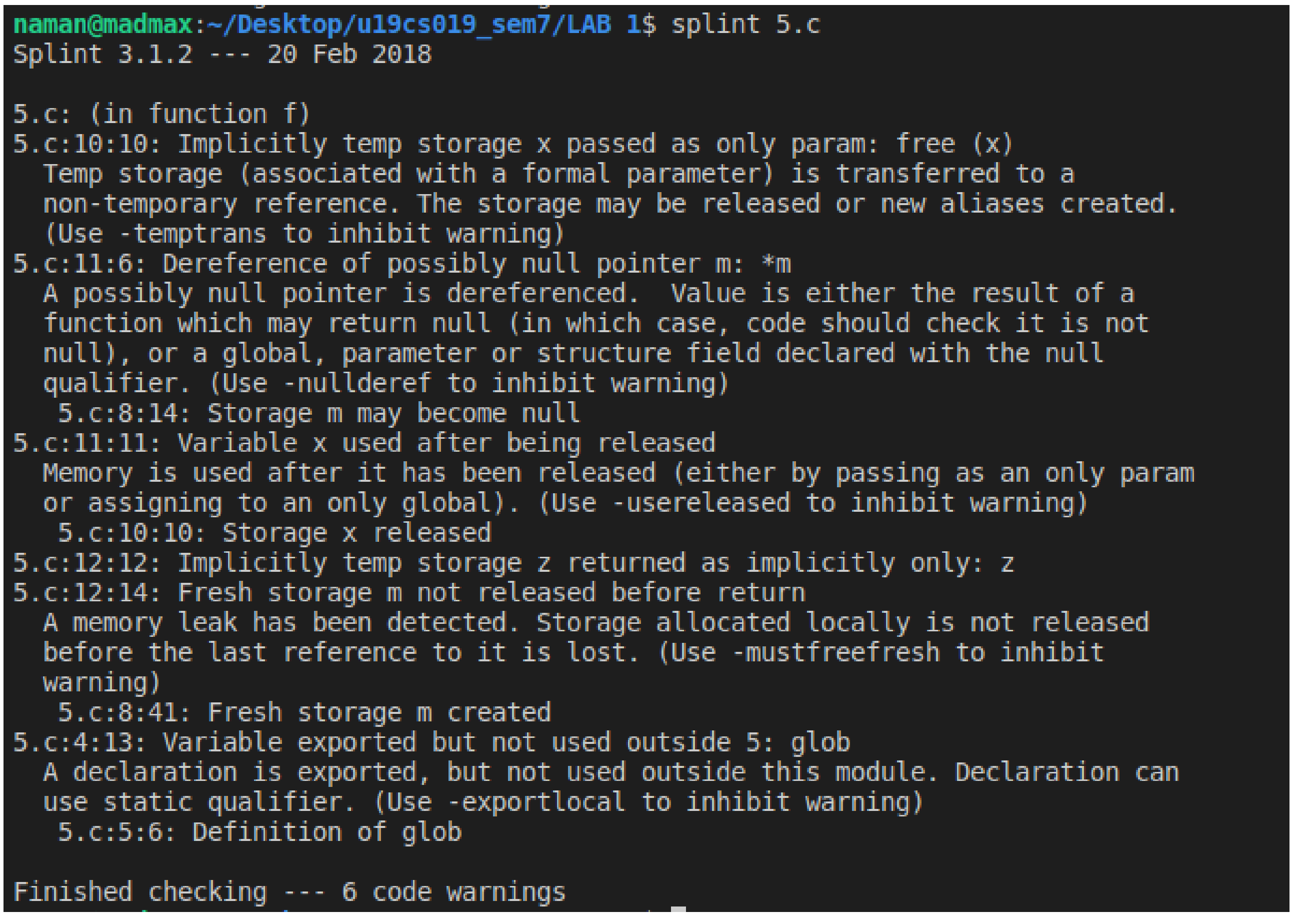
```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <string.h>
#include"str.h"

// typedef char *str;

bool isPalindrome(str 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()
{
    // callPal();
    return 0;
}
```



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 4.c
Splint 3.1.2 --- 20 Feb 2018

4.c: (in function isPalindrome)
4.c:11:29: Cast from underlying abstract type str: (char *)s
  An abstraction barrier is broken. If necessary, use /*@access <type>@*/ to
  allow access to an abstract type. (Use -abstract to inhibit warning)
4.c:12:30: Function strlen expects arg 1 to be char * gets str: s
  Underlying types match, but str is an abstract type that is not accessible
  here.
4.c:15:27: Array fetch from non-array (str): s[len - i - 1]
  Types are incompatible. (Use -type to inhibit warning)
4.c: (in function callPal)
4.c:22:26: Function isPalindrome expects arg 1 to be str gets char *: "bob"
  Underlying types match, but str is an abstract type that is not accessible
  here.
4.c:9:6: Function exported but not used outside 4: isPalindrome
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   4.c:19:1: Definition of isPalindrome

Finished checking --- 5 code warnings
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$
```

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

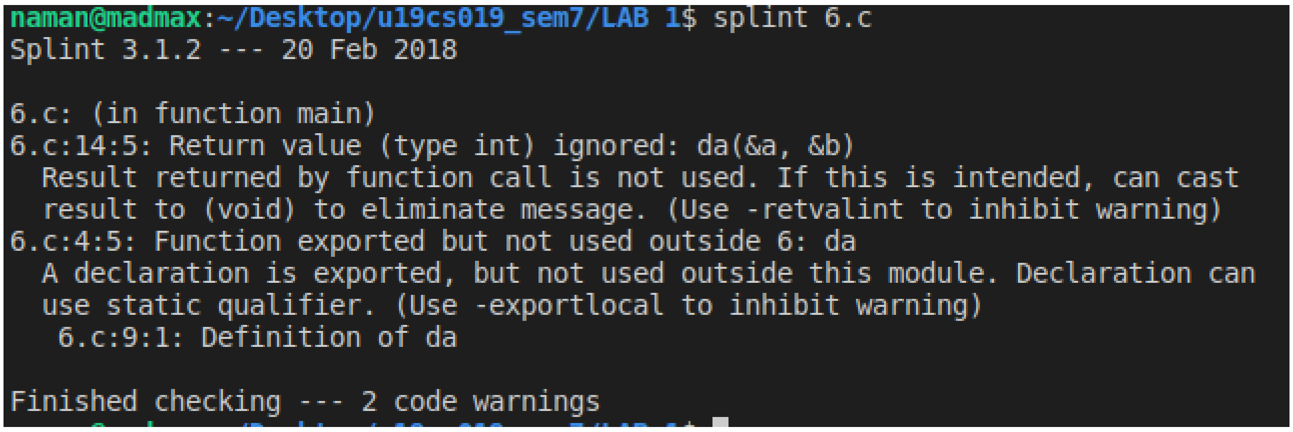
```
#include <stdio.h>
#include <stdlib.h>

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()
{
   return 0;
}
```



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 5.c
Splint 3.1.2 --- 20 Feb 2018

5.c: (in function f)
5.c:10:10: 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)
5.c:11:6: 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)
   5.c:8:14: Storage m may become null
5.c:11:11: 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)
   5.c:10:10: Storage x released
5.c:12:12: Implicitly temp storage z returned as implicitly only: z
5.c:12:14: 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)
   5.c:8:41: Fresh storage m created
5.c:4:13: Variable exported but not used outside 5: glob
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   5.c:5:6: Definition of glob

Finished checking --- 6 code warnings
```

6. Dangerous aliasing.

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

int da(int *ptr1, int *ptr2)
{
   *ptr1 = 10;
   *ptr2 = 11;
   return *ptr1;
}

int main()
{
   int a = 10,b=1;
   da(&a, &b);
   return 0;
}
```



```
naman@madmax:~/Desktop/u19cs019_sem7/LAB 1$ splint 6.c
Splint 3.1.2 --- 20 Feb 2018

6.c: (in function main)
6.c:14:5: Return value (type int) ignored: da(&a, &b)
  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)
6.c:4:5: Function exported but not used outside 6: da
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   6.c:9:1: Definition of da

Finished checking --- 2 code warnings
```