```
cg87-user19@instance-1:~/wed_26_3/dpmsm$ splint dpm.c
Splint 3.1.2 --- 21 Feb 2021

dpm.c:45:84: Comment starts inside comment
  A comment open sequence (/*) appears within a comment.  This usually means an
  earlier comment was not closed. (Use -nestcomment to inhibit warning)
dpm.c:162:92: Comment starts inside comment
dpm.c:252:107: Comment starts inside comment
patient.c: (in function patient_registration)
patient.c:20:8: Test expression for while not boolean, type int: 1
  Test expression type is not boolean or int. (Use -predboolint to inhibit
  warning)
patient.c:23:3: Return value (type int) ignored: scanf("%s", str)
  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)
patient.c:25:3: Assignment of size_t to int: len = strlen(str)
  To allow arbitrary integral types to match any integral type, use
  +matchanyintegral.
patient.c:26:6: Test expression for if not boolean, type int:
                  isdigit_validation1(str)
patient.c:40:20: Left operand of && is non-boolean (patient *):
                   (ptr1) && ptr1->aadhar_no != new->aadhar_no
  The operand of a boolean operator is not a boolean. Use +ptrnegate to allow !
  to be used on pointers. (Use -boolops to inhibit warning)
patient.c:43:44: Format argument 1 to printf (%s) expects char * gets int:
                    new->aadhar_no
  Type of parameter is not consistent with corresponding code in format string.
  (Use -formattype to inhibit warning)
   patient.c:43:26: Corresponding format code
patient.c:55:8: Test expression for while not boolean, type int: 1
patient.c:58:3: Return value (type int) ignored: scanf("%s", new-...
patient.c:60:6: Operands of > have incompatible types (size_t, int):
                  strlen(new->name) > max_len
patient.c:65:6: Test expression for if not boolean, type int:
                  isalpha_validation1(new->name)
patient.c:73:8: Test expression for while not boolean, type int: 1
patient.c:76:3: Return value (type int) ignored: scanf("%s", str)
patient.c:78:3: Assignment of size_t to int: a = strlen(str)
patient.c:79:6: Test expression for if not boolean, type int:
                  isdigit_validation1(str)
```

```
patient.c:99:3: Return value (type int) ignored: scanf("%s", str)
patient.c:101:3: Assignment of size_t to int: w = strlen(str)
patient.c:102:6: Test expression for if not boolean, type int:
                    isdigit_validation1(str)
patient.c:119:8: Test expression for while not boolean, type int: 1
patient.c:122:3: Return value (type int) ignored: scanf("%s", new-...
patient.c:123:6: Test expression for if not boolean, type int:
                    isalpha_validation1(new->name)
patient.c:136:3: Unreachable code: break
  This code will never be reached on any possible execution. (Use -unreachable
  to inhibit warning)
patient.c:138:8: Test expression for while not boolean, type int: 1
patient.c:141:3: Return value (type int) ignored: scanf("%s", new-...
patient.c:143:3: Assignment of size_t to int: len = strlen(new->phone_no)
patient.c:149:6: Test expression for if not boolean, type int:
                    isdigit_validation1(new->phone_no)
patient.c:169:3: Implicitly only storage new->next (type struct patient_details
                    *) not released before assignment: new->next = start1
  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)
patient.c:169:3: Unqualified storage start1 assigned to implicitly only:
                    new->next = start1
  Unqualified storage is transferred in an inconsistent way. (Use
  -unqualifiedtrans to inhibit warning)
patient.c:174:19: Left operand of && is non-boolean (patient *):
                     (ptr1) && ptr1->aadhar_no != new->aadhar_no
patient.c:177:2: Variable start1 is kept in false branch, but not kept in true
                    branch.
  The state of a variable is different depending on which branch is taken. This
  means no annotation can sensibly be applied to the storage. (Use -branchstate
  to inhibit warning)
   patient.c:177:2: in false branch:
   patient.c:176:3: Storage start1 becomes kept
   patient.c:177:2: in true branch:
   patient.c:170:3: Storage start1 becomes owned
patient.c:183:10: Parse Error. (For help on parse errors, see splint -help
            parseerrors.)
*** Cannot continue.
```