ENEE 457
Static Analysis Class Exercise

Consider the following code snippet on which we would like to perform a taint analysis. Type qualifiers are represented by capital letters: A, B, C, D, E.

```
 1  int printf(A char *fmt, ..);
 2  B char *fgets(..);
 3
 4
 5  int main () {
 6          C char *mystring = fgets(.., network_fd);
 7          D char *mystring2 = mystring;
 8          E char *mystring3 = ``Hello World'';
 9          mystring2 = mystring3;
10          printf(mystring2);
11          return 0;
12  }
```

i. Identify all the sources and sinks in the code snippet and determine the corresponding settings for the type qualifiers.

ln 1 (sink) A = untainted
ln 2 (source) B = tainted

ii. List all of the constraints on the type qualifiers.

ln 6 C >= tainted
ln 7 D >= C
ln 8 E >= untainted
ln 9 D >= E
ln 10 D < = untainted

iii. Is there a vulnerability in the above code? Is there a solution for the undetermined type qualifiers that satisfies all the constraints? If there is no vulnerability and no solution, it means that our taint analysis has produced a false positive. How can the taint analysis be modified so that the false positive is removed?

No vulnerability.
No solution since constraints from ln 6, ln 7, ln 10 imply:
tainted <= C <= D <= untainted
This implies that tainted <= untainted, which is false, since we assume tainted > untainted
When mystring2 is assigned in ln 9, it should be given a new name (each variable should only be assigned once). ln 9 becomes:
F char *mystring4 = mystring3
ln 10 becomes: printf(mystring4)
Now all constraints can be satisfied.