**Homework #3: Due Oct 9 (Thurs) 23:59 (200 points).**

1. (90 pts) To write down kcov-branch-identify using Clang based on the provided template C++ file. kcov-branch-identify receives a file name of a *preprocessed* single C file and prints the list of the branches and the total number of branches of the C file. You have to submit your kcov-branch-identify code both in hardcopy and softcopy.

See the following output for the attached example.c:

verifier3:~/kcov/branch-identify$ PrintBranches example.c

function: f1

If ID: 0 Line: 7 Col: 2

function: main

If ID: 1 Line: 18 Col: 2

If ID: 2 Line: 20 Col: 9

For ID: 3 Line: 27 Col: 2

While ID: 4 Line: 32 Col: 2

Do ID: 5 Line: 37 Col: 2

Case ID: 6 Line: 39 Col: 4

Case ID: 7 Line: 42 Col: 4

?: ID: 8 Line: 43 Col: 9

Default ID: 9 Line: 46 Col: 4

ImpDef. ID: 10 Line: 51 Col: 2

Case ID: 11 Line: 52 Col: 3

Case ID: 12 Line: 55 Col: 3

Total number of branches: 20

Note 1. We count each case as one branch (i.e., considering switch(){…} has multiple outgoing edges). Also, we count (implicit) default statement as one branch regardless of whether default exist or not. A line and a column of an implicit default is those of corresponding switch().

Note 2. You do not have to worry about header files neither macros since a target C file is a preprocessed file.

1. (10 pts) To print out the branches in the *preprocessed* grep v1.2 C file (i.e., grep\_pre.c) by using your kcov-branch-identify. Submit the entire output only in softcopy to save papers. Just report total # of branches in your HW hardcopy.

Note 1. A preprocessed grep C file can be obtained by gcc –E grep.c > grep\_pre.c

Note 2. You can ignore various Clang warnings.

Note 3. The total # of branches of the preprocessed grep C file > 3000

1. (90 pts) To write down kcov using Clang. You have to submit your kcov code both in hardcopy and softcopy.
   1. kcov receives a file name of a *preprocessed* single C file <f>.c and generates the instrumented version <f>-cov.c to measure branch coverage of <f>.c through testing.
   2. When <f>-cov.c is compiled and executed 1st time, <f>-cov.c generates a coverage measurement file <f>-cov-measure.txt. After then, <f>-cov.c updates <f>-cov-measure.txt through testing <f>-cov.c. The format of <f>-cov-measure.txt is as follows

Line# |# of execution |# of execution | conditional

|of then branch |of else branch | expression

1453 0 0 errnum

1474 0 7 size && !result

1484 3 0 ptr

1488 0 0 size && !result

…

6950 0 0 (end = memchr(beg + len, '\n', (buf + size) - (beg + len))) != 0

6955 0 0 beg > buf && beg[-1] != '\n'

Covered: 581 / Total: 3101 = 18.735892%

Note1. If one line has multiple branches (i.e., nested if statements), you can print out these branches in separate lines with the same line id

Note2. The # of execution of else branch of case should be always 0 (i.e., meaningless)

Note3. A conditional expression of default is ``default’’

1. (10 pts) To print out the coverage measurement file of the preprocessed grep C code with the following test cases where grep.c is the grep v1.2 C file (not preprocessed C file).

Submit the output only in softcopy to save papers. Just report the last line of the measurement file (i.e., # of covered branch, # of total branch, and achieved branch coverage percentage) in your HW hardcopy.

./grep –n "if" grep.c  
./grep –E "[0-9][0-9]+" grep.c  
./grep -E “[[:digit:]][[:alpha:]]” grep.c