Berkeley Programming Contest Fall 2017

Yi Wu, and P. N. Hilfinger

Please make sure your electronic registration is up to date, and that it contains the correct account you are going to be using to submit solutions (we connect names with accounts using the registration data).

To set up your account, execute

```
source ~ctest/bin/setup
```

in all shells that you are using. (We assume you are running bash. Others will have to examine this file and do the equivalent for their shells.)

This booklet should contain **??** problems on **??** pages. You have 5 hours in which to solve as many of them as possible. Each program must reside entirely in a single file. In Java, the class containing the main program for problem N must be named `P`N. Do *not* make class `P`N public (an artifact of how we process it). Each C/C++ file should start with the line

```
#include "contest.h"
```

and must contain no other `#include` directives, except as indicated below. Upon completion, each C/C++ and Java program *must* terminate by calling `exit(0)` (or `System.exit(0)` in Java).

Aside from files in the standard system libraries and those we supply (none this year), you may not use any pre-existing computer-readable files to supply source or object code; you must type in everything yourself. Selected portions of the standard `g++` class library are included among of the standard libraries you may use: specifically, for C++:

```
cstdlib cstdio climits stdarg.h cstring cctype iostream
iomanip string sstream vector list stack queue map set
bitset algorithm cmath unordered_map unordered_set
```

In Java, you may use the standard packages `java.lang`, `java.io`, `java.math`, `java.text`, and `java.util` and their subpackages. You may not use utilities such as `yacc`, `bison`, `lex`, or `flex` to produce programs. Your programs may not create other processes (as with the `system`, `popen`, `fork`, or `exec` series of calls or their Java-library equivalents). You may use any inanimate reference materials you desire, but no people. You can be disqualified for breaking these rules.

To submit a solution, go to our contest announcement page:

`http://inst.cs.berkeley.edu/~ctest/contest`

and click on the "web interface" link. You will go to a page from which you can upload and submit files from your local computer (at home or in the labs). On this page, you can also find out your score, and look at error logs from failed submissions.

You will be penalized for incorrect submissions that get past the simple test administered by `submit`, so be sure to test your programs (if you get a message from `submit` saying that it failed, you will *not* be penalized). All tests (for any language) will use the compilation command

`contest-gcc` N

followed by one or more execution tests of the form (Bourne shell):

`./`N `<` *test-input-file* `>` *test-output-file* `2>` *junk-file*

which sends normal output to *test-output-file* and error output to *junk-file*. The output from running each input file is then compared with a standard output file, or tested by a program in cases where the output is not unique. In this comparison, leading and trailing blanks are ignored and sequences of blanks are compressed to single blanks. Otherwise, the comparison is literal; be sure to follow the output formats *exactly*. It will do no good to argue about how trivially your program's output differs from what is expected; you'd be arguing with a program. Make sure that the last line of output ends with a newline. Your program must not send any output to `stderr`; that is, the temporary file *junk-file* must be empty at the end of execution. Each test is subject to a time limit (on ashby.cs) specified in the problem statement. You can see the test results using the web interface described above.

In the actual ACM contests, you will not be given nearly as much information about errors in your submissions as you receive here. Indeed, it may occur to you to simply take the results you get back from our automated judge and rewrite your program to print them out verbatim when your program receives the corresponding input. Be warned that I will feel free to fail any submission in which I find this sort of hanky-panky going on (retroactively, if need be).

The command `contest-gcc` N, where N is the number of a problem, is available to you for developing and testing your solutions. It converts your source program—N`.cc`, N`.java`, or N`.py`—into an executable program called just N.

For C and C++ programs, it is roughly equivalent to

`g++ -std=gnu++11 -Wall -o` N `-O3 -g -I`*our-includes* N`.* -lm`

for C/C++. (We compile C with C++; it works most of the time.) The *our-includes* directory (typically `~ctest/include`) contains `contest.h` for C/C++, which also supplies the standard header files.

For Java programs, it is equivalent to

```
javac -g -classpath .:our-classes N.java
```

followed by a command that creates an executable file called N that runs the command

```
java  PN
```

when executed (so that it makes the execution of Java programs look the same as execution of C/C++ programs).

For Python 3 programs, `contest-gcc` simply copies your file to N, makes it executable, and puts a line `#!/usr/bin/env python3` in front, which causes Unix to run it through the python3 interpreter.

The files in `~ctest/submission-tests/`N, where N is a problem number, contain the input files and standard output files that `submit` uses for its simple tests.

All input will be placed in `stdin`. You may assume that the input conforms to any restrictions in the problem statement; you need not check the input for correctness. Consequently, you C/C++ programmers are free to use `scanf` to read in numbers and strings and `gets` to read in lines.

Terminology. The terms *free format* and *free-format input* indicate that input numbers, words, or tokens are separated from each other by arbitrary whitespace characters. By standard C/UNIX convention, a whitespace character is a space, tab, return, newline, formfeed, or vertical tab character. A *word* or *token,* accordingly, is a sequence of non-whitespace characters delimited on each side by either whitespace or the beginning or end of the input file.

Scoring. Scoring will be according to the ACM Contest Rules. You will be ranked by the number of problems solved. Where two or more contestants complete the same number of problems, they will be ranked by the *total time* required for the problems solved. The total time is defined as the sum of the *time consumed* for each of the problems solved. The time consumed on a problem is the time elapsed between the start of the contest and successful submission, plus 20 minutes for each unsuccessful submission, and minus the time spent judging your entries. Unsuccessful submissions of problems that are not solved do not count. As a matter of strategy, you can derive from these rules that it is best to work on the problems in order of increasing expected completion time.

Protests. Should you disagree with the rejection of one of your problems, first prepare a file containing the explanation for your protest, and then use the `protest` command (without arguments). It will ask you for the problem number, the submission number (submission 1 is your first submission of a problem, 2 the second,

etc.), and the name of the file containing your explanation. Do not protest without first checking carefully; groundless protests will result in a 5-minute penalty (see Scoring above). The Judge will *not* answer technical questions about C, C++, Java, the compilers, the editor, the debugger, the shell, or the operating system.

Notices. During the contest, the Web page at URL

`http://inst.cs.berkeley.edu/~ctest/contest/index.html`

will contain any urgent announcements, plus a running scoreboard showing who has solved what problems. Sometimes, it is useful to see what problems others are solving, to give you a clue as to what is easy.