2025/2026 Southern California Regional ICPC On-Line Rehearsal
November 1, 2025 through November 13, 2025

The following steps are designed to be executed within the programming contest environment. This environment can be downloaded from the "Rehearsal VBox Appliance" link on the front page of https://scl.na.icpc.global/
Appliance login username: *icpcuser* Password: *icpcpw*

Warm-Up Problem 1

This problem should be completed first. Do all the steps before attempting problems 2 and 3.

The purpose of this problem is to familiarize all contestants with many parts of the environment. All contestants should submit this input correctly and run all commands listed before moving on to Warm-up Problems 2 and 3.

Open a browser. To connect to DOMjudge, connect to:

https://rehearsal.socalcontest.org/domjudge/

There is a shortcut for the "Terminal" for the command prompt at the bottom of the screen. IDEs can be reached from the command prompt, from the Applications menu under Development, or from shortcuts on the desktop.

Log in to DOMjudge using your team's DOMjudge credentials. (Note that at the actual contest, the credentials for Linux and DOMjudge will be the same.)

Step 1: Type in the problem code:

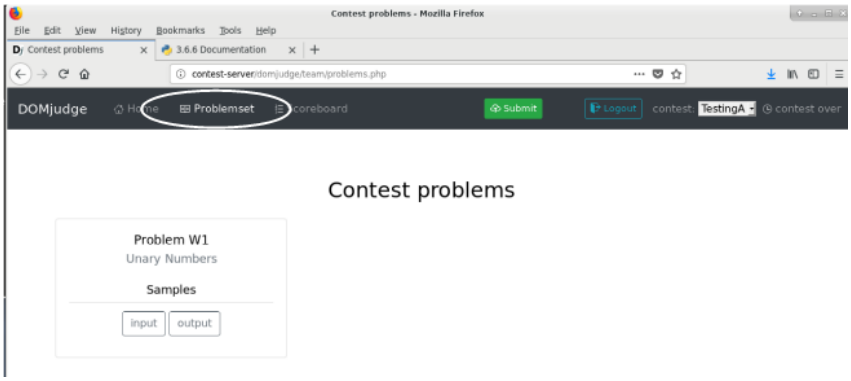
Select any one of the problem solutions and type it in (code follows) and save the file.

Step 2: Compile the code using from the command prompt:

compile *source_file*

Note that the compile command is not necessary for Python 3.

Step 3: Get any supplementary materials and sample input and output from DOMjudge:



Step 4: Test the code:

Use the test data provided, along with any other data you choose.

NOTE: During the contest your code will be judged against data you never see (the "judge's data"). The sample data provided are not exhaustive – it is your responsibility to design a thorough test plan.

Your program can be run after compilation with the following (compilation not needed for Python 3):

C, C++, input from keyboard `$ ./a.out`
C, C++, input from file *data.in* `$ ./a.out < data.in`
C, C++, input from *data.in*, output to *results.out* `$ ./a.out < data.in > results.out`

Java, input from keyboard `$ java classfile`
Java, input from file data.in `$ java classfile < data.in`
Java, input from data.in, output to results.out `$ java classfile < data.in > results.out`

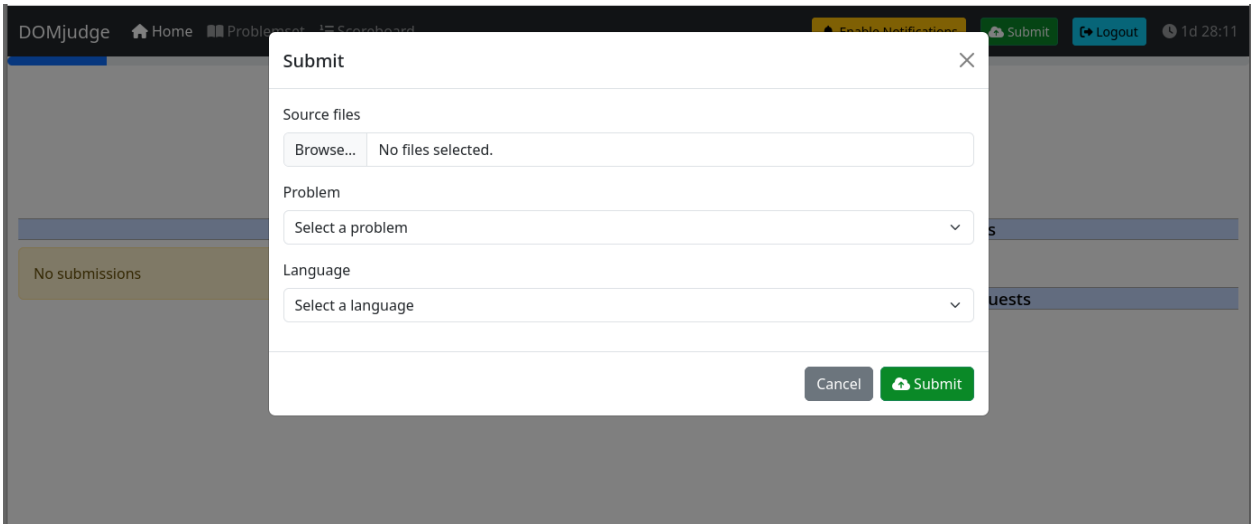
Python3, input from keyboard `$ python3 sourcefile.py3`
Python3, input from file data.in `$ python3 sourcefile.py3 < data.in`
Python3, input from data.in, output to results.out
`$ python3 sourcefile.py3 < data.in > results.out`

Kotlin, input from keyboard `$ kotlin ClassfileKt`
Kotlin, input from file data.in `$ kotlin ClassfileKt < data.in`
Kotlin, input from data.in, output to results.out
`$ kotlin ClassfileKt < data.in > results.out`

The judges recommend that you compare your program's output against the provided output from the problem sample input and output downloaded in Step 3. For example:

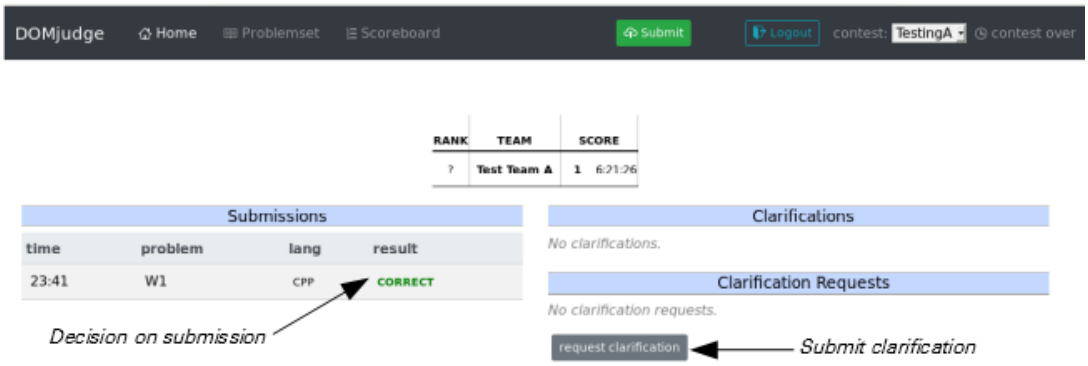
`$ diff results.out 1.ans`

Step 5: Submit the code via the DOMJudge interface, first select the source file, then click submit:



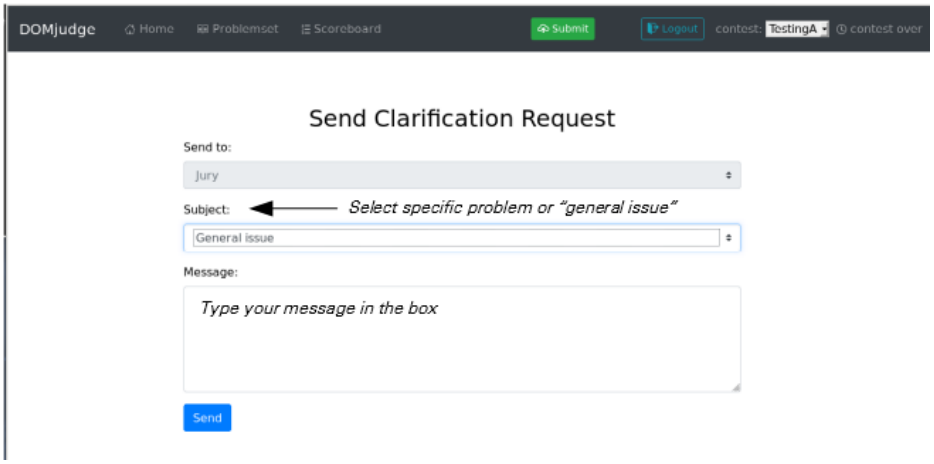
Kotlin submissions will also prompt for the entry point for your program. A default value of *ClassKt* (where the name of your class is substituted for *Class*) may be provided.

Step 6: See the results from your submission:



Step 7: Request a Clarification by clicking the *request clarification* button on the main page …

… and then completing the form. Use the problem number as the subject for questions on a specific problem.



If you don't have a specific Clarification regarding this rehearsal or one of the assigned problems, please request a Clarification to familiarize yourself with the process. Submit a question like "What was the color of that white horse?"

Step 8: (Optional) Find out how much time is left in the contest and look at the scoreboard:

Time left – Look at your start page from DOMjudge where it is displayed in the upper right-hand corner

Score –Click the *Scoreboard* button on the ribbon (see Step 3) to see the current scores; your team's score is displayed on the start page.

Step 9: See the on-line language/library documentation:

C++ library – file:///usr/share/doc/libstdc++-docs/html/index.html

Java API – file:///opt/docs/java/api/index.html

Python 3 – file:///opt/docs/python/index.html

Kotlin – file:///usr/share/doc/kotlin/kotlin-reference.pdf

Warm-up Problem from 2002/2003
Unary Numbers

What could be simpler than binary numbers? Unary numbers! A Unary number n, $n > 0$, is coded as $n - 1$ one bits followed by a zero bit. Thus the code for 5 is 11110. Here are some unary numbers.

decimal	*unary*
1	0
2	10
3	110
4	1110
5	11110
6	111110
7	1111110

Input consists of decimal numbers, one per line, with no leading or trailing whitespace. Each number will be in the range 1–76. Input is terminated by end-of-file.

For each number, produce a single line of output consisting of the input decimal number, with no leading zeroes or spaces, a single space, and the unary equivalent with no leading or trailing spaces.

Sample Input

```
76
37
5
28
14
8
1
```

Output for the Sample Input

```
76 11111111111111111111111111111111111111111111111111111111111111111111111111110
37 1111111111111111111111111111111110
5 11110
28 11111111111111111111111110
14 11111111111110
8 11111110
1 0
```

unary.cpp (or unary.cc)

```
#include <iostream>
using namespace std;

int main ()
{
   int n;

   while(cin >> n) {    // cin >> n is false when no values remain
      cout << n;
      cout << ' ';
      while (n > 1) {
         cout << '1';
         n--;
      }
      cout << '0' << endl;    // endl causes a newline, ASCII 0x0A, to be emitted
   }
   return 0;            // indicate normal program termination
}
```

unary.java:

```
import java.io.*;

class unary {                                //main class needs to match filename
   public static void main (String [] args) throws IOException
   {
      int            n;
      String         s;
      BufferedReader stdin;

      stdin=new BufferedReader(new InputStreamReader(System.in));
               // wrap BufferedReader around InputStreamReader around System.in

      while ( (s=stdin.readLine()) != null) {
               // BufferedReader.readLine returns null at end-of-file
         n=Integer.parseInt(s);
         System.out.print(n + " ");
         for (int i=n - 1; i > 0; i--) {
            System.out.print("1");
         }
         System.out.println("0");              // println() writes an ASCII 0x0A
      }
      System.exit(0);                       // indicate normal program termination
   }
}
```

---

unary.py3:

```
import sys

for line in sys.stdin:
    line=line.replace('\n','')  # remove end-of-line present in strings read from input
    n=int(line)
    print(n, end=' ')           # print number in decimal followed by one space
    for i in range(n - 1):
        print('1', end='')      # print '1' without any trailing characters
    print('0')                  # print '0' followed by newline

exit(0)                         # indicate normal program termination
```

unary.c:

```
#include <stdio.h>

int main()
{
   int  i;
   int  n;
   char s[4];  /* make room for up to two decimal digits, end-of-line (newline),
                  and a zero-byte to terminate the string */

   while(fgets(s,sizeof(s),stdin) != NULL) {
        /* read an entire line into s.  fgets() returns NULL at end-of-file */
      sscanf(s,"%d",&n);                      /* extract n from the input line */
      fprintf(stdout,"%d ",n);
      for (i=n - 1; i > 0; i--) {
         fputc('1',stdout);
      }
      fputs("0\n",stdout);
               /* the newline character, \n, emits an ASCII 0x0A */
   }
   return 0;                /* indicate normal program termination */
}
```

---

unary_seed.c:

```
#include <stdio.h>

int main()
{
   int  i;
   int  n;
   char s[4];  /* make room for up to two decimal digits, end-of-line (newline),
                    and a zero-byte to terminate the string */

/*
 * Attempt to read an entire line into s.  The read (fgets) preceding the while loop is the seed
read.
 */
   fgets(s,sizeof(s),stdin);               /* attempt to read an entire line into s */
   while( !feof(stdin) ) {                    /* while not end-of-file ... */
      sscanf(s,"%d",&n);                      /* extract n from the input line */
      fprintf(stdout,"%d ",n);
      for (i=n - 1; i > 0; i--) {
         fputc('1',stdout);
      }
      fputs("0\n",stdout);   /* the newline character, \n, emits an ASCII 0x0A */
      fgets(s,sizeof(s),stdin);          /* attempt to read an entire line into s */
   }
   return 0;                             /* indicate normal program termination */
}
```

unary.kt:

```
import kotlin.system.exitProcess

fun main() {
  var s: String?
  s=readLine()            // seed read
  while (s != null) {
    var n=s.toInt()
    print(n); print(" ")
    while (n > 1) {
      print("1");
      n -= 1;
    }
    println("0")
    s=readLine()
  }
  exitProcess(0)          // indicate normal program termination
}
```

2025/2026 SOUTHERN CALIFORNIA REGIONAL
INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Warm-Up Problem 2
Best of N

For a multi-game playoff series in a sport, a team must win the best of N games. At each point during the series, there is a minimum number of games that remain to be played. Sports venues need to be prepared to host at least the minimum number of games that are required. For example, during a best-of-seven series, with the standings of 1 win vs. 2 wins, a minimum of two games remain to be played.

Your team is to write a program that determines the minimum number of games remaining in a given series.

Input to your program consists of 1 to 12 lines, ending with end-of-file. Each line represents the status of a given series. Each line contains three integers separated by single spaces: N W_1 W_2, where N is the number of games in the best-of series (an odd number, $0 < N < 40$), W_1 is the current number of wins for Team 1 ($W_1 < \lceil N/2 \rceil$), and W_2 is the current number of wins for Team 2 ($W_2 < \lceil N/2 \rceil$).

For each input line, your program is to print a line giving the minimum number of games remaining as an integer.

Sample Input

```
7 1 2
7 0 0
3 1 1
```

Output for the Sample Input

```
2
4
1
```

2025/2026 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Warm-Up Problem 3
Intuitive Elements

Brandon is learning the periodic table! However, he doesn't like some of the elements because the symbol of the element contains letters which are not present in the name of the element. He finds this to be unintuitive, especially because in other contexts, he expects abbreviations to not introduce random letters.

Given a string and a proposed abbreviation, determine if Brandon would find it intuitive. Brandon finds an abbreviation intuitive if and only if every letter that appears in the abbreviation appears in the original string. Brandon does not look at the abbreviation carefully, so it is acceptable for a letter to appear more times in the abbreviation than in the original string, or for the letters to appear in a different order between the string and the abbreviation.

The first line of input contains a single integer t ($1 \leq t \leq 10^3$). This is the number of test cases. Each test case consists of two lines. The first line contains a single string a of length at most 50. This string only contains lowercase letters. The second line contains a single string b that is strictly shorter than a and also only contains lowercase letters. Neither string will be empty.

For each test case, if all the letters in b appear in a, print a line containing the string "`YES`". Otherwise, print a line containing the string "`NO`".

Sample Input

```
4
magnesium
mg
silver
ag
aabb
bbb
aabb
ba
```

Output for the Sample Input

```
YES
NO
YES
YES
```