SOUTHERN CALIFORNIA ICPC 2020-2021 WARMUP PACKET
revision 1.1.2

READ THIS FIRST!

The warmup session is especially important for the 2020-2021 online regional contest. Not only does it prepare you and the staff for the contest, it defines a set of reference information for use during the contest. It is possible that this warmup packet will change as problems are identified and fixed through the warmup process. Please pay attention to the bulletins and website (http://socalcontest.org) to download the latest revision of warmup packet. The warmup session establishes:

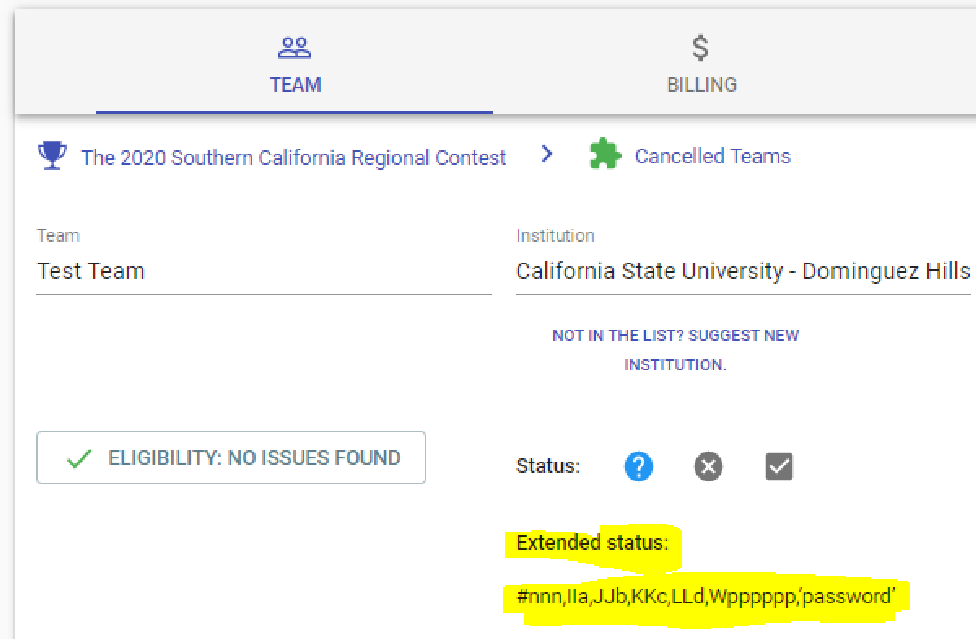
1. The method for getting Zoom meeting passcodes, and your Zoom and DOMJudge usernames & passwords.
2. Your team and each member's identity via a check-in process.
3. The ability for your computer to communicate with the Webinar and Zoom sessions.
4. The ability to communicate between team members.
5. The ability to communicate between you and the contest staff.
6. The ability for your computer to communicate with the DOMJudge contest server.

This document is a bit of a long read. Please take your time and read it thoroughly.

SOUTHERN CALIFORNIA ICPC 2020-2021 WARMUP PACKET
revision 1.1.2

1. Zoom meeting IDs & passcodes. Zoom and DOMJudge usernames and passwords.

Your information is displayed on your Team page. To access it, login to your Dashboard (at icpc.global). Expand the area labeled "Your Teams," then click on your team name. This is a link that will take you to your team screen. In the top section (see partial snip below) you will see "Extended Status" and a value.


TEAM
BILLING
The 2020 Southern California Regional Contest
Cancelled Teams
NOT IN THE LIST? SUGGEST NEW INSTITUTION.
ELIGIBILITY: NO ISSUES FOUND

Team
Test Team

Institution
California State University - Dominguez Hills

Status: ? ☒ ☑

Extended status:

#nnn,IIa,JJb,KKc,LLd,Wpppppp,'password'

The extended status contains an abbreviated set of information

#nnn,Wpppppp,'password'

where:

- nnn is your three-digit team number. Use this number to determine your DOMJudge username, of the form **icpc*nnn***. Use *password* (without the quotes) to log into the DOMJudge contest environment at https://rc2020-2021.socalcontest.org/domjudge. Each team member connects to DOMJudge with the same username and password.
- ***Once you read this paragraph***, work with your team to choose a unique letter, a, b, or c, for each of your team members. Substitute nnn, and your letter into ***α,*** in the address below to determine your Zoom login, of the form
  socal.icpc.zoom+*nnnα*@gmail.com
  Yes, that plus sign is part of the Zoom login. Use *password* (without the quotes) to log into Zoom. Please do not change the Zoom password. Please do not share the password outside your team.
- *pppppp* is the passcode for two different Zoom meeting IDs, the check-in and the warmup meetings. The Zoom meeting IDs for checking in and for the warmup session will be emailed to you with your assigned warmup session time.

- As the contest approaches, an additional string will be added, **C*pppppp***, with the passcode for the competition Zoom meeting. The password will remain the same.

For example, if your team members are Jane Doe, Bruce Lee, and Harvey D. Rabbit, and it was decided that Jane takes letter ‘a’, Bruce takes letter ‘b’, and Harvey takes letter ‘c’, the entry below

```
#056,W456123,’zazzle’
```

Leads to the following:

- The DOMJudge username is **icpc056**, password **zazzle**
- Jane Doe’s Zoom login is **socal.icpc.zoom+056a@gmail.com**, password **zazzle**
- Bruce Lee’s Zoom login is **socal.icpc.zoom+056b@gmail.com**, password **zazzle**
- Harvey D. Rabbit’s Zoom login is **socal.icpc.zoom+056c@gmail.com**, password **zazzle**
- For check-in, join the Zoom check-in meeting ID assigned to you, using passcode **456123**
- For the warmup session, leave the check-in meeting, then join the warmup Zoom meeting ID assigned to you, using passcode **456123**

2. Team Check-in.

The registrar will use the Zoom meeting id sent with your warmup session assignment. At the assigned check-in time for your team, please connect to Zoom using the passcode obtained from the extended status. When done checking in, leave the Zoom check-in meeting.

3. Connect to the Webinar and warmup Zoom sessions.

Follow the steps below to join:

- Connect to webcast using a modern web browser. The webinar URL will be posted to http://socalcontest.org/current/importantupdates.shtml. See Step 5.
- Connect to Zoom using your **socal.icpc.zoom+nnnα@gmail.com** login. It is recommended that you use the downloadable Zoom client (version 5.4 or higher) rather than a web browser. The Zoom client has better features than the browser interface. Upon entry to Zoom, you should be placed in a breakout room that contains only your team.

4. Communicate with your team members.

While in Zoom,

- Share your screen with your teammates.
- Send a chat to Everyone in your breakout room.
- Share a file with Everyone in your breakout room.
- If you have a microphone, speak with your teammates in your breakout room.
- If you have a camera, turn on video and smile for your teammates in your breakout room.

If your team uses other collaboration tools or services, use this opportunity to confirm that they work to your satisfaction.

5. Communicate with the contest staff.

There are many methods of communicating with the contest staff, with each method using a different communication channel in the event the failure of any other channel.

- Use DOMJudge's clarification feature. The clarification feature within DOMJudge is your first choice for communicating with the staff during the contest. After connecting to DOMJudge (see step 6), make sure you send at least one clarification request.
- Send an email to **`socal.icpc.help@gmail.com`**. If there are problems connecting to DOMJudge, there is a support email address that works independently of DOMJudge. You may send email to the staff from your preferred personal email address. Use your DOMJudge username as the subject line; the email support staff divides work by username ranges. Write your question or concern in the body of the email message.
  To: **`socal.icpc.help@gmail.com`**
  Subject: ***`yourDOMJudgeUsername`*** (example: `icpc056`)
  Body: **`please reply`**
  You may use email support to summon staff to your breakout room. Use a message body of 'request breakout room support'. When staff support is available, the Zoom host will direct a tech to your breakout.
- During the warmup, make a brief voice call (if feasible and the charges are not significant) to United States phone number **949-257-2378**. This is a last resort method for when your internet data communication channel fails. You might have to leave a voicemail. Leave your name, your team number, and your letter (a through d) designation.
- View the up-to-the-moment status webpage,
  http://socalcontest.org/current/importantupdates.shtml

6. Communicate with the DOMJudge contest server.

This is where your team uses your development environment or the virtual appliance to compete in the warmup contest.

- Use a web browser to connect to https://rc2020-2021.socalcontest.org/domjudge. Login using your DOMJudge username and password from step 1.
- Obtain the problem set using the Problemset link in DOMJudge's home page.
- Obtain a second copy of the problem set independently of DOMJudge from the hyperlink contained in http://socalcontest.org/current/importantupdates.shtml. This is a copy of the problem set, available should the DOMJudge server encounter problems. If the DOMJudge server becomes unavailable, you should use the time to work on solution development.
- Follow the guided tutorial below to learn the environment.

Warm-Up Problem 1—*Save this document for reference during the contest!*

This problem should be completed first. Do all the steps before attempting problem #2.

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 Problem 2.

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

https://rc2020-2021.socalcontest.org/domjudge

Within the virtual appliance, there is a shortcut for the "Terminal" for the command prompt at the bottom of the screen. IDEs such as Geany and Eclipse can be reached from the command prompt or from the Applications menu under Development.

Step 6.1: Type in the problem code:

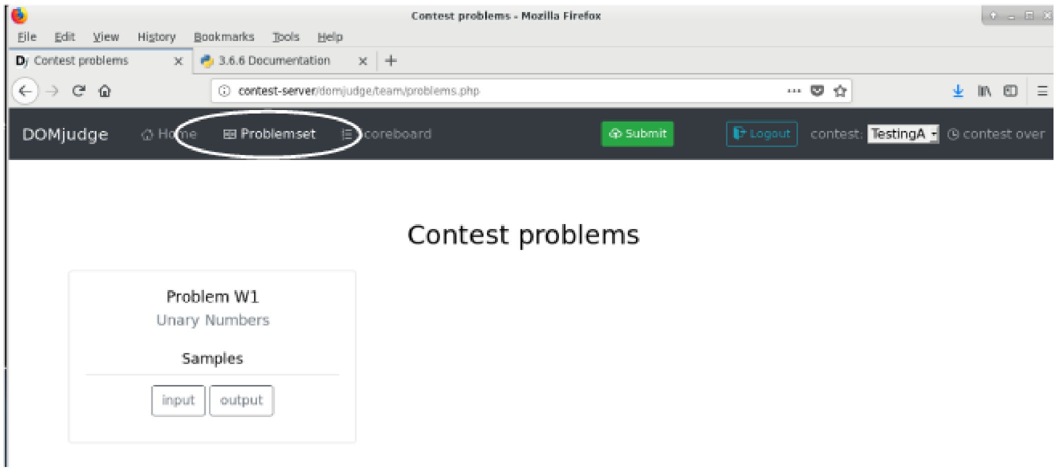
Select any one of the problem solutions and type it in (code follows) and save the file.

Step 6.2: Compile the code using from the command prompt (available in the virtual appliance):

compile *source_file*

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

Step 6.3: Get any supplementary materials and sample input and output from DOMjudge:

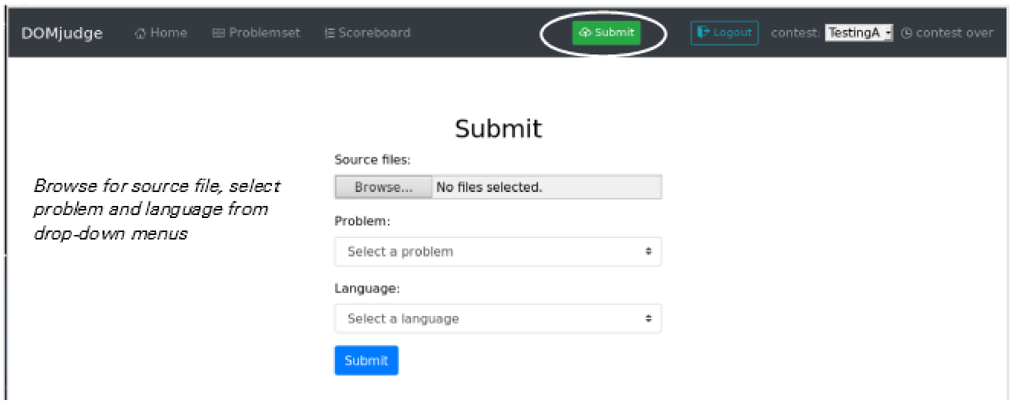


Step 6.4: Test the code:

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 is not exhaustive – it is your responsibility to design a thorough test plan.

Step 6.5: Submit the code via the DOMJudge interface, first select the source file, then click submit:



Submit

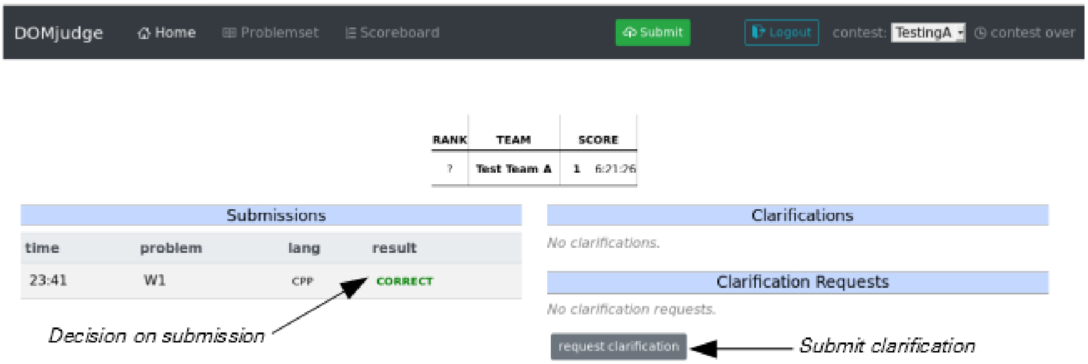
Browse for source file, select problem and language from drop-down menus

Source files: Browse... No files selected.

Problem: Select a problem

Language: Select a language

Step 6.6: See the results from your submission:



RANK	TEAM	SCORE	
?	Test Team A	1	6:21:26

Submissions

time	problem	lang	result
23:41	W1	CPP	CORRECT

Decision on submission

Clarifications

No clarifications.

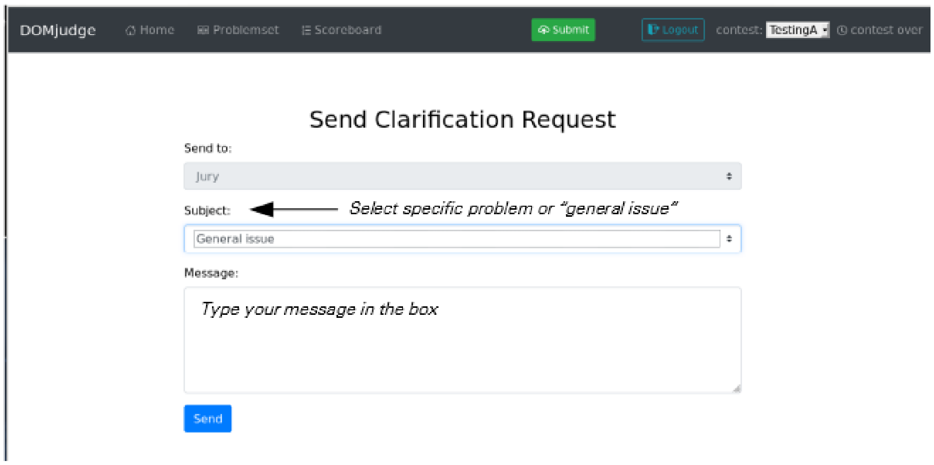
Clarification Requests

No clarification requests.

Submit clarification

Step 6.7: 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.



Send Clarification Request

Send to: Jury

Subject: General issue

Select specific problem or "general issue"

Message:

Type your message in the box

Step 6.8: (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.

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 1111111111111111111111111111111111111111111111111111111111111111111111111110
37 1111111111111111111111111111111111110
5 11110
28 1111111111111111111111111110
14 11111111111110
8 11111110
1 0
```

SOUTHERN CALIFORNIA ICPC 2020-2021 WARMUP PACKET
revision 1.1.2

unary.cpp (or unary.cc)

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

int main ()
{
   int n;

   cin >> n;            // seed read
   while(!cin.eof()) {  // eof() valid only after attempted read
      cout << n;
      cout << ' ';
      while (n > 1) {
         cout << '1';
         n--;
      }
      cout << "0\n";    // the newline character, \n, emits ASCII 0x0A
      cin >> n;
   }
   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 */
}
```