2022/2023 Southern California Regional ICPC Rehearsal/"Warm-Up"
February 18, 2023; 10:00 AM PST

Zoom Webinar ID: 839 5779 7197
https://us02web.zoom.us/j/83957797197

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 the Firefox 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 or from the Applications menu under Development.

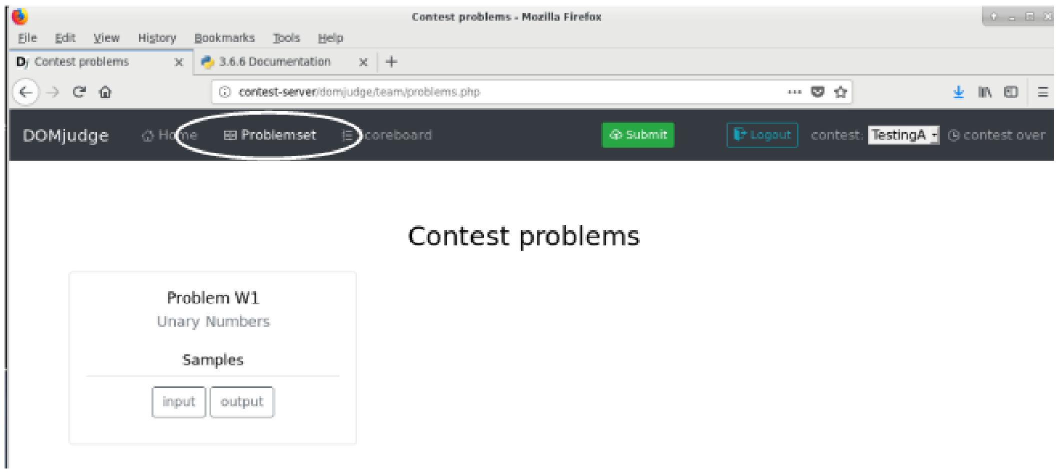
Step 1: Type in the problem code:
Select any one of the problem solutions and type it in (code follows) and save the file.

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:


Contest problems - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Contest problems
3.6.6 Documentation
contest-server/domjudge/team/problems.php
DOMjudge Home Problemset Scoreboard Submit Logout contest: TestingA contest over
Contest problems
Problem W1
Unary Numbers
Samples
input output

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):

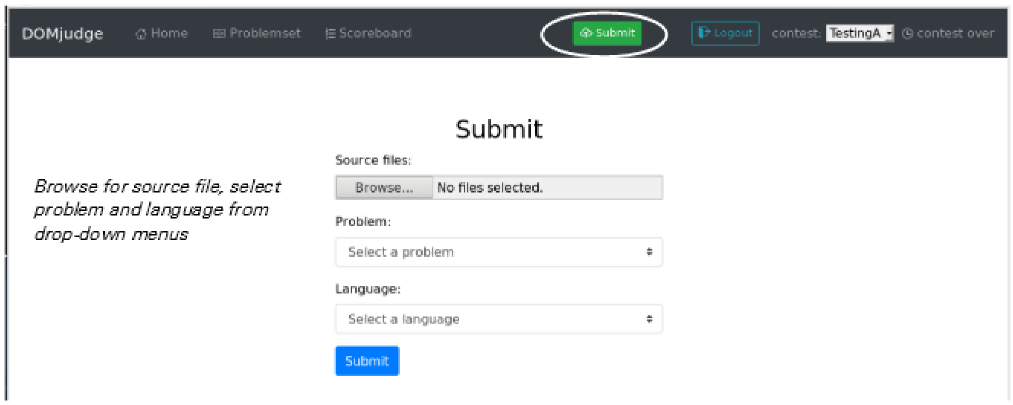
C, C++, input from keyboard `$ ./a.out`
C, C++, input from file *data.in* `$ ./a.out < data.in`
C, C++, input from *data.in*, output to *results.out* `$ ./a.out < data.in > results.out`

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

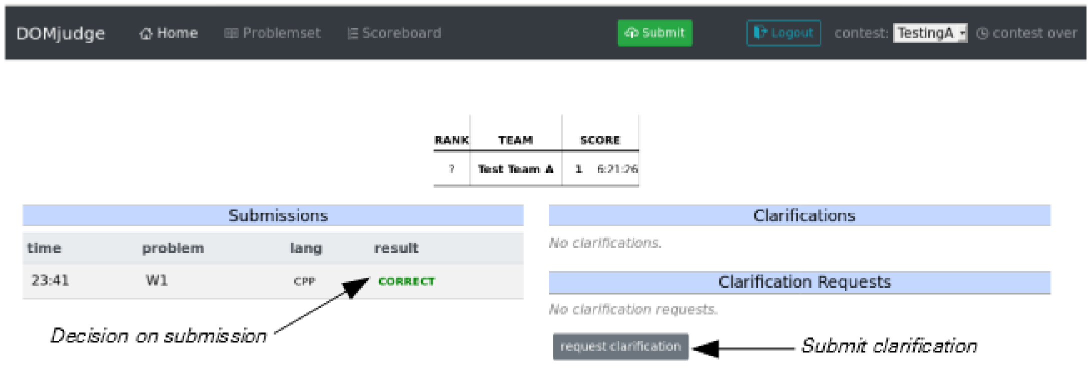
```
 $ python3 sourcefile.py3 < data.in > results.out
```

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



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

Step 6: See the results from your submission:


DOMjudge Home Problemset Scoreboard Submit Logout contest: TestingA contest over

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

Submissions

time	problem	lang	result
23:41	W1	CPP	CORRECT

Clarifications

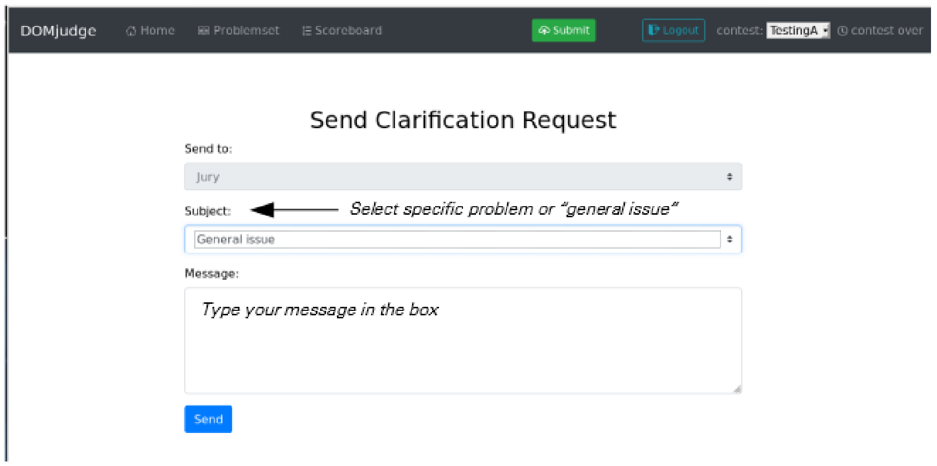
No clarifications.

Clarification Requests

No clarification requests.

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.


DOMjudge Home Problemset Scoreboard Submit Logout contest: TestingA contest over

Send Clarification Request

Send to: Jury

Subject: General issue

Message:

Select specific problem or "general issue"
Type your message in the box
Send

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?"

One objective of this rehearsal period is verifying that all components of the contest management system are working and configured correctly. The officials are reviewing everything. Submission response times are likely to be longer than they will be at the actual contest for similar submissions.

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 (in appliance V1.1)

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
```

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 */
}
```

2022/2023 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. 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, W_1 is the current number of wins for Team 1, and W_2 is the current number of wins for Team 2. $W_1, W_2 < \lceil N/2 \rceil$. $0 < N < 40$, where N is always an odd number.

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
```

2022/2023 SOUTHERN CALIFORNIA REGIONAL
INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Warm-Up Problem 3
Northwest by North

There are thirty-two standard points on a compass, evenly spaced around the circumference. They are depicted and named on a diagram known as the "compass rose." A compass rose is shown in Figure 1.

The circumference of the compass is also divided into 360 degrees, increasing clockwise from North. Due North is 0°, East is 90°, South is 180°, and West is 270°.

Your team is to write a program that will take a list of values in degrees and print the nearest standard compass point for each value.

Input to your program will be a series of one to twenty measurements in degrees, one per line, each starting in the first column. Each measurement M will be in the range $0° \leq M < 360°$ and may contain up to two digits after the decimal point. The last measurement will be followed by the end-of-file.

For each measurement, print a line containing the nearest standard compass point as it appears around the edge of the compass rose in Figure 1.

Sample Input

```
0
270
155.5
178.91
326
```

Output for the Sample Input

```
N
W
SSE
S
NWbN
```

Warm-Up Problem 3
Northwest by North (continued)

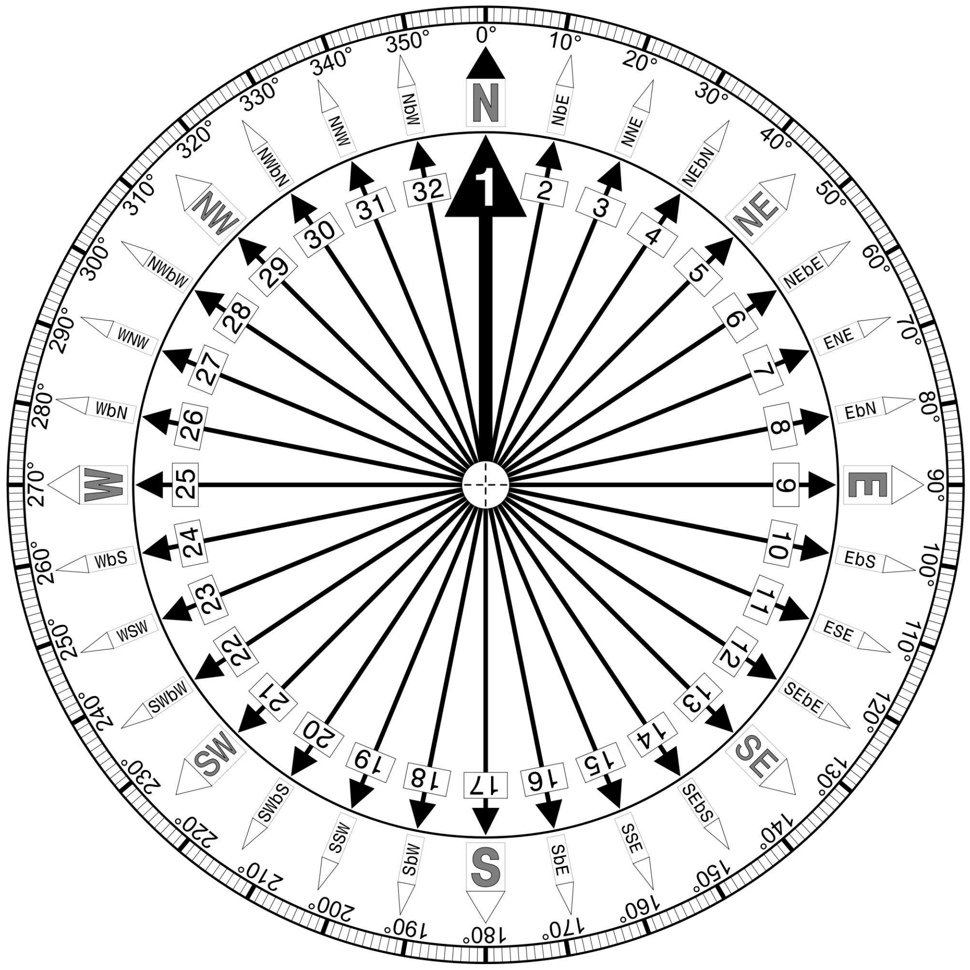


Figure 1. Compass Rose showing the thirty-two standard compass points.