Introduction

MATA Hack 4 Good: High School Programming Competition

Saturday November 4th 2017 9:00am-1:00pm CDT
Organizer: Mid-America Technology Alliance (MATA)
Venue: The eFactory, 405 N Jefferson Ave, Springfield MO 65806

Welcome to the **MATA "Hack 4 Good" High School Programming Competition!** This event provides students with an opportunity to demonstrate and sharpen their programming skills. Continue to improve your programming skills by participating in other programming competitions.

Our High School competition **does** provide Internet access, Sample Solutions, and Hints. Please be aware that other competitions such as the ACM ICPC **do not** allow Internet access or help of any kind.

High School Programming Competitions

- SBU HSPC (facebook.com/sbucis/) - 11/17/2017 at SBU in **Bolivar MO (30 miles)***
- UoA HSPC (hspc.csce.uark.edu) - March 2018 at UoA in **Fayetteville AR (120 miles)***
- Google Code-In (codein.withgoogle.com) - 11/28/2017
- Google Summer of Code (developers.google.com/open-source/gsoc/) - Summer 2018
- USACO (usaco.org) - Online Qualifying Contests, International Contest
- ACSL (acsl.org) - Local Qualifying Contests, National Contest

College Programming Competitions

- CCSC-CPRPC (ccsc.org/centralplains/) 04/06/2018 at NMSU in Maryville MO
- ACM-ICPC (icpc.baylor.edu)
- HP CodeWars (hpcodewars.org)
- IEEE Xtreme (ieeextreme.org) - Annual Online Challenge
- Google Code Jam (code.google.com/codejam/) - Online Contests, International Contest

* Information about this event is attached to your packet.

Special thank you to Sherry Coker, Greg Johnson, Robin Robertson, Jason Klein, Thomas Rankin, Maranda Provance, Myke Bates, and Fredrick Lawler for their help organizing and hosting this event!

Event Schedule

9:00am	9:30am	Team Registration and Check-In
9:30am	10:00am	Team Orientation (Welcome, Review Rules, Submit Practice Problems)
10:00am	12:00pm	Programming Competition
12:00pm	12:30pm	Lunch
12:30pm	1:00pm	Awards

Event Rules

The following **MATA "Hack 4 Good" High School Programming Competition** event rules have been adapted from the ACM ICPC rules (http://acmacpc.org/rules/).

General

1. Contestants (and reserve contestant) must wear the official contest T-shirt during the contest.
2. Contestants may only use one computer per team during the contest. Teams may bring spare computers for use if primary computer fails. Computer should include a web browser and development tools they are familiar with for Java, Python 2, Python 3, C (C11), or C++ (C++14).
3. Contestants may bring printed material and reference books to the contest hall.
4. Electronic devices (mobile phones, tablets, calculators) are strictly prohibited during the contest.
5. Bags brought into the contest area may be checked.
6. Contestants are not allowed to talk to their coaches during the contest. In general, contestants are not to converse with anyone except members of their team and personnel designated by the contest director (e.g. for help submitting problems to the scoring system).
7. A team may be disqualified by the contest director for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, or distracting behavior.
8. Contestants are not allowed to open problem statements until directed by the contest director.

Contest Format

1. The contest will last exactly two hours (unless there is an unforeseen difficulty that requires extending the time.)
2. The contest problem-set is made of 12 problems. Each team attempts to solve as many problems as possible using a single computer.
3. The allowed programming languages are: Java, Python 2, Python 3, C (C11), and C++ (C++14). Contestants may use any of these allowed languages to solve each problem.
4. A contestant may submit a claim of ambiguity or error in a problem statement by submitting a clarification request. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants. Notification of accepted runs may be suspended to keep the final results secret. Notification of rejected runs will continue until the end of the contest.
5. Solutions are judged by running them against secret test cases. The contest judges are solely responsible for determining the correctness of the submitted solutions and their decision is final.
6. Rejected runs are classified by the scoring system as follows:
    - **Compile Error**: The compiler returned an error when compiling the program.
    - **Runtime Error**: The program crashed due to segmentation fault, division by zero, etc.
    - **Time Limit Exceeded**: The program failed to complete and terminate within 5 seconds.
    - **Wrong Answer**: The program terminated successfully, but the output was incorrect.
    - **No Output**: The program terminated successfully, but there was no output.
7. If a run is rejected, the team is free to try again and send as many runs as they wish until the problem is solved (but make sure you understand the scoring rules).

Problem Format

1. Every effort is made to guarantee that problems avoid dependence on detailed knowledge of a particular application area or particular programming language.
2. Each problem will require reading input from Standard Input (STDIN), and printing its output to the Standard Output (STDOUT). (Coaches: Make sure at least one team member can read data from STDIN and write data to STDOUT. Refer to Sample Solutions for each Practice Problem.)
3. Since judging is an automated process, it is mandatory for the program output to match the output format specified in the problem description.
4. The judges' secret input files will test the program on multiple cases (or datasets). The format of the input file will be designed so that multiple datasets can be included in a single text file.
5. Each problem in the contest is specified in the following sections:
    - **Problem Header:** Estimated Time, Number of Points, and Source Code Filename
    - **Description**: This section specifies the problem contestants are supposed to solve and also provides description on how the input file will be formatted as well as the format of the output of your program. Your program's output must conform to the format specified.
    - **Sample Input/Output**: Here you'll find a sample Input/Output that has successfully passed the judges' program. Your program will be tested on a different (and more complex) dataset. Just because your program passed the sample Input/Output, doesn't mean it is correct.
    - **Learn More**: Additional information about this problem, or about the sample inputs.
6. In addition, the problem statement will specify the name you should use for your program.
7. Since compilation and testing is an automated process, it is important that you follow the naming convention exactly. For example, if the problem statement states that the filename should be "prob01" then the program file must be named "prob01.java" for a **Java** program, "prob01.py2" for **Python 2**, "prob01.py3" for **Python 3**, "prob01.c" for **C**, or "prob01.cpp" for **C++**.

Programming Your Solutions

1. Each program must be in a single source file with the name specified in the problem description; failure to meet this requirement will result in a "Syntax or Compilation Error".
   Note for Java programmers: You can have more than one top-level class in a single source file. Only the main class should be declared public. All other classes should be unqualified (i.e. just "class", without "public").
2. Output will be judged using a file comparison utility. Output must be exactly as specified in the output format section: Spelling, punctuation, spacing, and case (uppercase/lowercase) are all significant (unless specified otherwise in the problem statement.)
3. Your program cannot require any intervention from the user. All input must be read from Standard Input (STDIN). If you submit a program that requires user intervention, you will likely receive a "Time Limit Exceeded" error. Contestants must NOT place a "Press any key to continue" statement at the end of their program.
4. All test cases used in judging will conform to the input specifications. **Your program does NOT need to detect invalid input** such as incorrect data types or missing values.
5. Make sure your program will compile and run locally before you upload your solution.

Contest Scoring

1. Teams are ranked in a descending order according to the number of points they accumulate.
2. Teams who accumulate same number of points are ranked ascendingly by total time (the score).
3. The team who is first to solve a problem will receive special recognition on the scoreboard.
4. If the solution a team provides is incorrect, the team will be penalized 5 minutes for their incorrect attempt to solve the problem.
5. The total time (aka score) is the sum of the time consumed for each problem solved.
6. The time consumed for a solved problem is the time elapsed from the beginning of contest to the submittal of the accepted run; plus 5 penalty minutes for every rejected run of that problem.
7. No time is consumed for a problem that is not solved (even if there are rejected runs for it.)

Scoring Example

Consider the following team which submitted 6 runs: three for problem A, two for problem B, and one for C. For each submitted run, the table shows the time of the submission, for which problem, and the judge's' response for that particular run. For solving problem A, the time consumed is 17 plus 2*5 for the two unsuccessful runs. For solving problem B, the time consumed is 0 since both runs were unsuccessful. For solving problem C, the time consumed is 13 minutes. So the total time for this team 17+2*5+0+13 = 40. (Notice that no penalties were added for problem B since it wasn't solved correctly.)

Resolving Ties

Teams solving the same number of problems with the same total time are ranked by the geometric mean of the individual times for each solved problem (smaller being better) without the penalties. Any remaining ties are left unbroken.

Judges' Decisions

Judges are solely responsible for accepting or rejecting submitted runs. In consultation with the judges, the contest director determines the winners of the contest. The contest director and judges are empowered to adjust for or adjudicate unforeseen events and conditions. Their decisions are final.

Eligibility

1. Each student must be currently enrolled in a high school (or equivalent program) in the Southwest Missouri region or the surrounding regions.
2. Contestants compete in teams of 1-3 students.

Awards and Recognition

- One trophy per team for the top 1st/2nd/3rd place teams.
- One balloon per team for each problem solved. Search online for "ICPC balloons" to learn more.
- Scoreboard will recognize which team is First To Solve each problem.
- Competition Results will be published online.

Important Instructions

Read the following instructions carefully. They contain important information about how to submit your solutions for automated scoring. If you have any questions about these instructions, ask a volunteer before the contest begins.

Program Input/Output

All program input must be read from Standard Input (STDIN) and all program output must be written to the screen via Standard Output (STDOUT).

Test your programs by saving the program input in a text file, then redirect the contents of your text file to your program at runtime. For example, a file named “problem.txt” can be redirected to STDIN of your program using syntax similar to this:

```
$ java problem < problem.txt
$ python problem.py < problem.txt
$ problem.exe < problem.txt
```

Submitting Programs

You must submit a **single source code file** for your program to the scoring website. The scoring website will NOT accept multiple source code files or compiled/executable files.

Carefully test your program to make sure it is using the inputs to generate the desired output. The scoring website will automatically compile and test your solution using different inputs.

Scoring Tips

Each problem is assigned a specific number of points and estimated time to complete. More difficult problems are assigned more points than less difficult problems. See Problem Difficulty chart below.

Your team will gain special recognition on the scoreboard if you are first to submit a correct solution for a problem. Solutions are scored in the order they are received.

Your team will be penalized 5 minutes if you submit an incorrect solution. Carefully test your solution before uploading to the scoring system!

Problem Difficulty

Practice Problems 1-2: Complete as Group (10 points each)
Problems 1-4: Beginner (10 minutes each, 100 points each)
Problems 5-8: Intermediate (15 minutes each, 150 points each)
Problems 9-12: Advanced (20 minutes each, 200 points each)

Practice Problems

The following problems are designed to ensure that each team is able to submit a test program to the scoring website, and that your program is able to read input (STDIN) from the scoring website, and that the scoring website is able to read output (STDOUT) from your program.

Practice Problem 1: Testing Output

5 minutes, 10 points
Filename: prac01 *(e.g. prac01.c, prac01.cpp, prac01.java, prac01.py2, prac01.py3)*

Description
This problem will ensure that the scoring website is able to read output (STDOUT) from your program. Your program does not need to read input for this problem.

Output a single line containing the message “Hello, World!”. Be sure your output includes the **exact** same **case**, **punctuation**, and **spacing** shown in the sample output. Only use a single space unless instructions specifically tell you otherwise.

Sample Input

```
None
```

Sample Output

```
Hello, World!
```

Hints
Need help writing to Standard Output? See the Sample Solutions for this problem on the next page.

Learn More
While small test programs have existed since the development of programmable computers, the tradition of using the phrase "Hello, world!" as a test message was influenced by an example program in the seminal book The C Programming Language.

The example program from that book prints "hello, world" (without capital letters or exclamation mark), and was inherited from a 1974 Bell Laboratories internal memorandum by Brian Kernighan, Programming in C.A Tutorial:

```
#include <stdio.h>

main( )
{
        printf("hello, world\n");
}
```

https://en.wikipedia.org/wiki/”Hello,_World!”_program
Prepared by Jason Klein. Adapted from CodeWars.

Practice Problem 1: Sample Solutions

See below for **Practice Problem 1** solutions written in **Java, Python 2, Python 3, C, and C++**.

Each solution demonstrates how to output strings to Standard Output (STDOUT). **Each solution has been tested with our scoring system.**

Each solution also demonstrates how to compile the program and how to read input from a text file during runtime. Each program will generate the **Sample Output** shown below.

Sample Output

```
Hello, World!
```

Sample Java Code (prac01.java)

```
import java.io.*;

public class prac01 {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Compile, Run

```
javac prac01.java
java prac01
```

Sample Python2 Code (prac01.py2)

```
print "Hello, World!"
```

Run

```
python2 prac01.py2
```

Sample Python3 Code (prac01.py3)

```
print("Hello, World!")
```

Run

```
python3 prac01.py3
```

Check Python Version

```
python -V
```

This will output the exact version, such as “Python 2.7.13” for **Python 2** or “Python 3.5.3” for **Python 3**.

Sample C Code (prac01.c)

```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

We recommend using the C11 standard. Be sure to return 0 in your code. A missing or different return statement might result in invalid exit codes that the scoring system interprets as a Runtime Error.

Compile, Run

```
gcc prac01.c -o prac01
./prac01
```

Sample C++ Code (prac01.cpp)

```
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

We recommend using the C++14 standard. Be sure to return 0 in your code. A missing or different return statement might result in invalid exit codes that the scoring system interprets as a Runtime Error.

Compile, Run

```
g++ prac01.cpp -o prac01
./prac01
```

Practice Problem 2: Testing Input and Output

5 minutes, 10 points
Filename: prac02 *(e.g. prac02.c, prac02.cpp, prac02.java, prac02.py2, prac02.py3)*

Description
This problem will ensure that your program is able to read input (STDIN) from the scoring website, and that the scoring website is able to read output (STDOUT) from your program.

The first line of input will include a single integer that indicates how many additional lines of input need to be read. Each additional line of input will contain two words separated by a space.

For each line of input containing NOUN and VERB, output a sentence constructed as “NOUN is VERB today!” Be sure that your output includes the **punctuation** shown in the sample output.

Sample Input

```
3
Mitchell coding
Danese testing
Larry debugging
```

Sample Output

```
Mitchell is coding today!
Danese is testing today!
Larry is debugging today!
```

Hints
Need help writing to Standard Output? See the Sample Solutions for this problem on the next page.

Learn More
Mitchell Baker was instrumental in the creation of the Mozilla Foundation, the organization that oversees development of the Firefox browser and other products. Today, she is chairperson of both the Mozilla Foundation and Mozilla Corporation.

Danese Cooper currently serves as chairperson of the Node.js Foundation. She also works in an open source role at PayPal. Previously, she served as CTO of the Wikimedia Foundation.

Larry Augustin is CEO of SugarCRM. He helped coin the term “Open Source”. In 1993 he founded VA Linux (now SourceForge, NASDAQ:LNUX), where he served as CEO until 2002.

Practice Problem 2: Sample Solutions

See below for **Practice Problem 2** solutions written in **Java, Python 2, Python 3, C, and C++**.

Each solution demonstrates how to read integers and strings from Standard Input (STDIN) and output strings to Standard Output (STDOUT). **Each solution has been tested with our scoring system.**

Each solution also demonstrates how to compile the program and how to read input from a text file during runtime. When using the following **Input**, each program will generate the following **Output**.

Sample Input (prac02.in)

```
3
Mitchell coding
Danese testing
Larry debugging
```

Sample Output

```
Mitchell is coding today!
Danese is testing today!
Larry is debugging today!
```

Sample Java Code (prac02.java)

```
import java.io.*;

public class prac02 {
    public static void main(String[] args) throws IOException {
        BufferedReader reader
            = new BufferedReader(new InputStreamReader(System.in));

        int numCases = Integer.parseInt(reader.readLine());

        while (numCases > 0) {
            --numCases;
            String line[] = reader.readLine().trim().split("\\s+");
            String noun = line[0];
            String verb = line[1];
            System.out.println(noun + " is " + verb + " today!");
        }
    }
}
```

Compile, Run with Input File

```
javac prac02.java
java prac02 < prac02.in
```

Sample Python2 Code (prac02.py2)

```
import sys

lines = sys.stdin.readlines()
num_cases = int(lines[0])

case = 1
while case < num_cases + 1:
    noun, verb = lines[case].split()
    print "%s is %s today!" % (noun, verb)
    case = case + 1
```

Run with Input File

```
python2 prac02.py2 < prac02.in
```

Sample Python3 Code (prac02.py3)

```
import sys

lines = sys.stdin.readlines()
num_cases = int(lines[0])

case = 1
while case < num_cases + 1:
    noun, verb = lines[case].split()
    print("%s is %s today!" % (noun, verb))
    case = case + 1
```

Run with Input File

```
python3 prac02.py3 < prac02.in
```

Sample C Code (prac02.c)

```
#include <stdio.h>

int main(int argc, char **argv) {
    int numCases;
    char noun[99 + 1]; /* + '\0' */
    char verb[99 + 1]; /* + '\0' */

    scanf("%d", &numCases);
    while (numCases) {
        --numCases;
        scanf("%s %s", noun, verb);
        printf("%s is %s today!\n", noun, verb);
    }
    return 0;
}
```

We recommend using the C11 standard. Be sure to return 0 in your code. A missing or different return statement might result in invalid exit codes that the scoring system interprets as a Runtime Error.

Compile, Run with Input File

```
gcc prac02.c -o prac02
./prac02 < prac02.in
```

Sample C++ Code (prac02.cpp)

```
#include <iostream>
#include <string>

using namespace std;

int main(int argc, char **argv) {
    int numCases;
    string name, verb;

    cin >> numCases;
    while (numCases) {
        --numCases;
        cin >> name >> verb;
        cout << name << " is " << verb << " today!" << endl;
    }
    return 0;
}
```

We recommend using the C++14 standard. Be sure to return 0 in your code. A missing or different return statement might result in invalid exit codes that the scoring system interprets as a Runtime Error.

Compile, Run with Input File

```
g++ prac02.cpp -o prac02
./prac02 < prac02.in
```