2016 UF High School Programming Contest
Contest Guide

Rules and Scoring

The rules for this contest follow that of the ACM ICPC. For a detailed list of rules refer to "https://icpc.baylor.edu/regionals/rules". The contest will be four hours in length and consist of 8-12 problems. Students will write programs according to problem specifications and submit solutions through the contest environment (see how to do so below). Each team has a *score* and *penalty points*. Each solved problem contributes +1 to a team's score. Teams are ranked by their score with higher scores being better than lower scores. For teams with the same scores, ties are broken with their penalty points; teams with lower penalty are preferred over teams with higher penalty. When a team solves a problem, the time elapsed (in minutes) since the start of the contest is added to their penalty; additionally, each submission marked as incorrect will add +20 to a team's penalty (these penalty points will only be added when the team finally solves the problem). Team's may have paper reference material with them for the duration of the contest. Teams may not talk to other teams during the contest. If a team is found talking with another, they will be warned once; after that, the team may be disqualified.

Contest Environment

The contest environment is based on the DOMjudge contest system. To access the contest environment, open up firefox and type "**http://uf.hspc/**". Make sure to verify that you are logged into the correct team account during the practice contest! If you have any questions about the problem statements, *use the clarification system*. Don't ask volunteers questions about the problem statements! All other questions should be directed to volunteers. When a solution is submitted it is sent to the judge host which will, after a brief intermission, return one of these potential verdicts:

1. *Accepted* - Congratulations! You can move on to the next problem.
2. *Incorrect* - This means that the output your program produced when given the judges input data did not match the correct judge output. +20 penalty points will be added to your score (after the problem is solved by the team).
3. *Compiler Error* - Your program did not compile correctly on the judge system.
4. *Runtime Error* - Your program threw an exception (this could potentially be a stack overflow in Java!). If you're coding in C++ then this could be a segmentation fault.
5. *Time Limit Exceeded* - Your program took too long to run on the judge system. +20 penalty points will be added to your score (after the problem is solved by the team).

Some things to consider

1. Problems are judged for correctness *and* speed. Solutions that are too slow will receive a *time limit exceeded* verdict (note that an incorrect submission may still be judged as time limit exceeded). Time limits are problem dependent.
2. Some problems may require the use of large data types. Be cautious with which ones you use. An example of this is found in the practice contest (longs are required!).
3. If you are using Java, be careful with how you use Scanner. Make sure you are able to read test data correctly. The *count the vowels* problem in the practice contest will be a good test of a team's efficacy with Scanner.
4. Pay attention to the input bounds! Some input will be very large in order to indicate that you need to write efficient code (sometimes a clever idea is required)!
5. Some problems may require rounding! We don't really like to penalize teams for not knowing how to do this so we'll give a hint: use printf.
6. If you have questions about the problem statements use the clarification system. On firefox, select the 'Team' bookmark and look for the "Submit Clarification" button on this page. For questions that need to be answered immediately, please flag down a volunteer by raising your hand (don't ask volunteers any questions about the problem set, please).
7. **Data should be read from *standard input* and written to *standard output*. In C++ this is cin/cout, respectively. In Java this is System.in/System.out, respectively.**

Java Solution to *count*

```
import java.util.*;
public class count {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int cases = sc.nextInt();
        sc.nextLine();//nextInt doesn't remove '\n'
        for (int i = 0; i < cases; i++) {
            String line = sc.nextLine().toLowerCase();
            int cnt = 0;
            for (char ch : line.toCharArray()) {
                if (ch == 'a' || ch == 'e'
                    || ch == 'i' || ch == 'o' || ch == 'u') {
                    cnt++;
                }
            }
            System.out.println(cnt);
        }
    }
}
```

C++11 Solution to *count*

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string line;
    getline(cin, line);
    int t = stoi(line);
    for (int i = 0; i < t; i++) {
        string line;
        getline(cin, line);
        int vowels = 0;
        for (int i = 0; i < line.size(); i++) {
            if (line[i] == 'a' || line[i] == 'A' ||
                line[i] == 'e' || line[i] == 'E' ||
                line[i] == 'i' || line[i] == 'I' ||
                line[i] == 'o' || line[i] == 'O' ||
                line[i] == 'u' || line[i] == 'U')
                vowels++;
        }
        cout << vowels << endl;
    }
    return 0;
}
```

Python 2 Solution to *count*

```
numLines = int(raw_input())
while numLines:
    numLines -= 1
    currLine = raw_input().lower()
    total = 0
    for i in currLine:
        if i in 'aeiou':
            total += 1
    print total
```