ILLINOIS TECHNOLOGY ASSOCIATION

ITA TECH CHALLENGE

CODE / COMPETE / CONNECT 2014

THE FINAL CHALLENGE

Good Luck on the Challenge!
You are the future of Illinois Tech.

PRESENTING SPONSOR

CDK Global™

Welcome to the 2014 ITA Tech Challenge!

We recommend skimming through the challenge prior to beginning in order to get an idea of the format and questions. In general, you'll be acquiring and manipulating some data to produce an output file that you'll be submitting to the judges.

See the final page of the exam for instructions on how to submit your answers.

Good Luck!

A. Get Your Data (maximum 25 points)

In order to proceed with the exam, you'll need some data. In order to acquire this data, you must ask for it from a web service. The web service is available at:

http://2014.fallchallenge.org

Task

Your task is to write code that executes an HTTP request to the above server, and capture the return data either into memory or into a file. The data itself is not terribly large (approximately 115 KB) so don't worry about bandwidth or disk space.

In order to get the web service to respond, you must set a specific HTTP header as part of your request, otherwise an HTTP `401 Unauthorized` error response will be returned. The header is:

```
X-TechChallege: true
```

You can do either of the following:

A. Use a tool, browser, or command line utility to download the data and store it (5 points)

   Or

B. Write code to execute an HTTP request with the required setup to download and capture the data (25 points)

B. Extract Your Data (Maximum 25 points)

The data you've downloaded from the web service in Part A is a compressed file, and therefore must be decompressed before you can continue. The compression scheme used was the standard GZip algorithm.

Task

Your task is to decompress the raw data downloaded from the web service. You can do either of the following:

A. Decompress the data using a command line tool or other utility (5 points)

   Or

B. Write code that decompresses the data using a third party API or framework (25 points)

C. Visualize Your Data (Maximum 50 points)

Your decompressed data from Part B is a flat ASCII text file containing a set of 3D points and their associated "faces", where a face is defined as a triangle made up of 3 points. Each point or face is represented on its own line in the text file, with each line being delimited with a single `\n` character.

A single 3D point/vertex is represented as follows:

```
point X, Y, Z
```

where X, Y, and Z are the point's position in 3D space within the unit cube. In the file, each encountered point is assigned an index, starting at index 0. Each point seen in the file is assigned an index, in order. In this way, the file contains a set of points indexed [0…n].

A single face is represented as follows:

```
face P0i, P1i, P2i
```

where P0i, P1i, and P2i are the indexes of 3 points from the set of points. Therefore each face is a triangle, and the file contains a set of such triangles.

Task

Your task is to visualize the 3D data. You should submit a PNG image that meets the following criteria:

A. The image should be 1024x1024 pixels and have a black background.

B. The unit square in 2D space is mapped to the image:
   a. Pixel `(0,0)` in the upper left of the image maps to X, Y coordinate `(-1,1)`
   b. Pixel `(1024,1024)` in the lower right of the image maps to X, Y coordinate `(1,-1)`
   c. Pixel `(512,512)` in the center of the image maps to the origin at X, Y coordinate `(0,0)`

C. You can use a simple perspective 3D projection with scaling to create 2D points from 3D points:

$$x_p = \frac{5x}{1/(z-2)} \qquad y_p = \frac{5y}{1/(z-2)}$$

D. If a point transforms to outside the image's 1024x1024 display, simply ignore it.

To visualize the data, perform any or all of the following:

A. Display a non-black pixel or other shape per 2D point (20 points)

B. Apply two 3D rotational transforms of each point prior to display. You can rotate 15 degrees about the X axis, and -30 degrees about the Y axis for a decent view. (20 points)

C. Display a line for each face present in the file, in addition to the pixel or other shape you're displaying in (A) (10 points)

Some Notes

1. Format and comment your code! Well formatted, documented code will help our graders to understand what your thought processes were, and may help them to award partial credit. And in general, when working in a professional environment on a team, good code documentation is a must. Show us you know how to do it. This means:
    a. Documenting your functions with parameters and return types, exceptions thrown, etc
    b. Documenting your logic, especially around non-intuitive or complex portions
    c. Including any other documentation/comments that would aid another developer in understanding your code
    d. Not over-documenting, either.

2. More efficient resource (CPU, memory, etc) usage will score more highly than less efficient usage. In most cases, an O(ln(n)) algorithm will be graded higher than an O(n), which will in turn be graded higher than an $O(n_2)$. If making a tradeoff between time and memory, give priority to time.

3. Don't reinvent the wheel. Libraries and frameworks exist for a reason. In a professional environment developers must often think about how to most efficiently spend their time, and often this means using components or features of 3rd party libraries. If you're trying to write a PNG encoder as part of solving this problem, you're probably not approaching the things correctly. *With that said, if you're sure you've received full credit on the rest of the exam with time to spare, and can manage to write a PNG or JPEG encoder in the time we've given you, we'd be impressed.*

4. Some environments/languages will make things easier than others, so choose the right tools for the job. Some languages that are likely to work well are: Java, C#, PHP, and Python. C++ can work too. However, these are just suggestions. If you're proficient with another language and can implement everything, then feel free to use it.

Submitting Your Exam

To submit your exam, you should collect your source files and your output PNG file and name them as follows:

1. For code, use the following scheme:

   `[last name]-[first name]-[school]-part[Part].ext`

   For example, if you are John Doe from UIUC, then your C++ code for part A might be named:

   `doe-john-uiuc-partA.h`

   and

   `doe-john-uiuc-partA.cpp`

2. If you wrote multiple parts in the same file or program, feel free to combine them as follows:

   `doe-john-uiuc-partA-partB-partC.java`

3. Don't send us project files, solution files, project metadata, make files, build scripts, or binaries.

4. If we don't receive code for a part but you submit answers for subsequent parts, we'll assume you used a tool or other third party utility and score you in the lower score bracket for that part.

5. For your output PNG, use a similar naming scheme:

   `doe-john-uiuc-final-image.png`

6. Make every effort to name your submissions properly and accurately. If we can't find your answer or figure out who you are, you probably won't receive credit for your submission.

7. Email your answers with the appropriate files attached to the following email address:

   `techchallenge@illinoistech.org`

   The timestamp of the email on our mailserver will serve as the authoritative clock for ties in score when we choose a winner.