Project 1 - CGI

Submit Assignment

Due Sep 20 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip
Available Sep 6 at 12am - Sep 22 at 11:59pm 17 days

Project #1 - Due Wednesday-September 20, 23:59:59pm

Use Canvas and submit a ZIP file as "Project #1".

Note: You shall follow Dr. Finkel's checklist for good programming:
http://www.cs.uky.edu/~raphael/checklist.html

Xelkalai from the planet Xarnaco has travelled through a wormhole
and recently arrived here. During the transport through the wormhole
his ship's computer was damaged. He is having trouble with both
navigation and communications. He is attempting to return home but needs
your help. He has commissioned you to write a universal web app to
make the needed translation/calculations. He will then manually
input the answers from your web app into his system.

- You are to implement a simple web page to perform simple conversions,
noted below.

- Your web page must be implemented on the www.cs.uky.edu webserver.

- Your web page must reside at the following URL:
http://www.cs.uky.edu/~YOURUSERID/CS316/xelk.html

- Your CGI program must reside at the following URL:
http://www.cs.uky.edu/~YOURUSERID/CS316/do_xelk.cgi

- Your HTML file shall:
- use an HTML form to ask the user for (4) parameters:
  1) original units
  2) new units
  3) amount of original units
  4) conversion factor (default value=1.0)

- either allow the user to type the units, or use a <select> input
- amount of original units can be a floating point number
- use the GET method and send the user input via the ACTION attribute
  to your CGI program (see below).
- The parameter names shall be:
origunits, convunits, numunits, convfactor - all lowercase!
This is important for grading! If you choose different names
then Xelkalai's form that he wrote will not work correctly with
your CGI program.

- Your CGI program shall:
- be written in perl, python, ruby, or C/C++.
- never abort due to incorrect user input.
- validate inputs submitted.
- shall perform the requested conversion to convert the amount in

parameter 3 from the units in parameter 1 to parameter 2. Due to weird gravitational and/or temporal variances, Xelkalai may provide a conversion factor. Use this value to multiply by the answer you calculate. In your form, default this value to 1.0.
- return the converted amount back to the user or an appropriate error message.
- NOT call any external URLs from within the program.
- be properly documented.

Your HTML/CGI files should handle the following conversions:
(Note, that your program should handle conversions from the same original units as the new units, as the conversion factor might be used!)

parsec to lightyear (1 parsec = 3.26 lightyears)
lightyear to kilometer (1 lightyear = 3.086 x 10^13 kilometers)
xlarn to parsec (1 xlarn = 7.3672 parsecs)

galacticyear to terrestrialyear (1 galacticyear = 250,000,000 terrestrialyears)
xarnyear to terrestrialyear (1 xarnyear = 1.2579 terrestrialyears)
terrestrialyear to terrestrialminute
(1 terrestrialyear = 525600 terrestrialminutes)

Your program should handle these 6 conversions and their reverses (for example: both xlarn to parsec AND parsec to xlarn)

Your program does NOT need to make indirect conversions like parsec to kilometer.

NOTE: those units (all lower case!) are the only units you should accept. Anything else should result in an error message. Also, if the amount is missing, generate an error message! If the amount or the conversion factor are not numbers, generate an error message.

Style:

- Your CGI program shall:
- output proper HTML
- contain a proper <style> tag to satisfy the following requirements
- output the body of the HTML two parts:
- the request (the 4 parameters submitted via the form)
- the answer
- if the request and answer are proper, output the request parameters in blue and the answer in green (again, using the appropriate tags and style settings.
- if the request includes improper parameters, output the offending parameter in red and in place of the answer, the error message in red with a bold font.

- minimize duplicate/complicated code. Think about a function or a data structure that would eliminate a large repetition of if/then/else statements.

What you are required to turn in your ZIP archive - note, do NOT create an additional directory in the ZIP archive - just these 3 files!

1. A file named "xelk.html".
2. A file named "do_xelk.cgi", or the source code (do_xelk.c) if C/C++.
3. A file named "project1.pdf" (not a word/doc/docx file!)

"project1.pdf" - external documentation shall contain the following:

Name(s)
A brief description of the project (1 or 2 lines)!

Answers to the following questions:

1) What happens when your do_xelk.cgi program has no parameters passed into it?

2) What happens if your do_xelk.cgi program is passed in just the origunits parameter and not the confunits parameter?

3) What happens if your do_xelk.cgi program is passed a string other than the 8 strings listed in the units?

4) What happens if your do_xelk.cgi program is given an empty string for the numunits parameter?

5) What happens if your do_xelk.cgi program is given an unknown parameter (say, for example, extra=Foo) as the first parameter?

BONUS: We could (should?) have used a POST method instead of GET method to send the user input in to our CGI program. Give 2 reasons why this might have been better than using GET, explicitly related to this assignment (even though its main objective is silly and not serious).
Hint: one is security, but why?

Program 1 - CGI Rubric		
Criteria	**Ratings**	**Pts**
HTML form view longer description		15.0 pts
External document view longer description		10.0 pts
CGI program view longer description		75.0 pts
		Total Points: 100.0