**Shell Programming - Points: 88 (13 of 25%)**

Write code for an executable shell script that will do the following actions, in the exact order given below. (You will write approximately 50 lines of executable code, plus a **step number** comment before each step.)

Summary and Purpose:

The first argument to the script should be a URL of a Unix **tar** file. The second argument is an optional program name. (The script will prompt the user for a missing second argument.) The script will fetch the URL and look for the program name in the **tar** file.

Only Step Comments are Required:

Put *only* a one-line comment containing the **step number** in front of the executable code in each step.

---

1. *[Points: 6]* Start your script with a standard NET2003 script header; but, do *not* include the Purpose or Assignment Label. (Remember the One-Line Description and Syntax.)
2. *[Points: 9]* If the number of arguments is incorrect, issue a good error message (follow the NET2003 guidelines) and exit the script with status 1.
3. *[Points: 1]* Put the first argument (the URL) into a variable named **`url`**.
4. *[Points: 12]* If the string in variable **`url`** begins with the seven characters **`http://`**, display **`Fetching UUU from web...`**, where **`UUU`** is the user's URL; otherwise, issue a good error message and exit the script with status 2.
5. *[Points: 2]* Put a unique temporary file name (located in the standard Unix temporary file directory) into a variable named **`tmp`**. The file name must contain the current process ID number.
6. *[Points: 10]* Fetch the raw (unformatted) URL contained in the **`url`** variable into the file named by the **`tmp`** variable. If the fetching fails with a bad return status, issue a good error message, remove the file, and exit the script with status 3.
7. *[Points: 9]* Make sure the fetched URL file is not empty (has a file size larger than zero); otherwise, issue a good error message, remove the file, and exit the script with status 4.
8. *[Points: 3]* Remove all write permissions from the the file containing the fetched URL. Do not change any other permissions. Quick-exit the script (no message) with status 5 if changing permissions fails.
9. *[Points: 10]* Run the **file** command on the fetched URL file and look for the string **`'tar archive'`**. If the fetched URL file is a **tar** archive (the string is found), display **`Looking in tar archive...`**; otherwise, issue a good error message and exit the script with status 6.
10. *[Points: 7]* If there are two arguments, put the second argument (the program name) into a variable named **`prog`**; otherwise, get the missing program name from the user and put it into variable **`prog`**. Quick-exit the script (no message) with status 7 if the user signals EOF to the script.
11. *[Points: 10]* Produce a table of contents (TofC) of the **tar** archive file and look in the TofC for the program name string contained in the **`prog`** variable. If the string is found in the **tar** archive TofC, display **`Program PPP is contained in UUU`**, where **`PPP`** is the program name found and **`UUU`** is the URL of the **tar** archive; otherwise, issue a good error message and exit the script with status 8.
12. *[Points: 7]* If the parent directory is writable and a file named **`foo.tar`** exists in the current directory, move the file to the parent directory. Quick-exit the script (no message) with status 9 on move failure.
13. *[Points: 2]* Rename the **tar** archive file to be **`foo.tar`**. Quick-exit the script (no message) with status 10 if the rename fails.

**Put a one-line comment containing the step number in front of the executable code in each step.**