CS 375 - UNIX System Programming
Fall 2018 – Project 4

Note: do not wait until the night before (or even two or three nights before) to start this project...

Assignment

Write a pair of C/C++ programs named **expand** and **factor**. **expand** should symbolically multiply out products of sums and exponentiated sums. **factor** is the inverse program; that is, it should find the factors of a given symbolic polynomial.

Here is an example expand session:

```
$ expand
> (x+1)^5
x^5+5*x^4+10*x^3+10*x^2+5*x+1
> (x+2*y)^2 * (x+4)^3
4*x^3*y^2+48*x^2*y^2+192*x*y^2+256*y^2+4*x^4*y+48*x^3*y
+192*x^2*y+256*x*y+x^5+12*x^4+48*x^3+64*x^2
> quit
$
```

In the above example **expand** is run with no arguments and so it goes it to a prompt-evaluate interaction loop with the user. (The expand prompt is just '**>**'. All user input is shown in bold.) It should read input from standard input and write output to standard output. When the user types "**quit**", the program should terminate.

If **expand** is run with one or more arguments (in quotes), it should display each argument, then an equal sign, then the expansion and then exit. Here is example output when **expand** is run with two arguments:

```
$ expand "(x + 3)^4" "(w + 2*z)^2"
(x + 3)^4 = x^4+12*x^3+54*x^2+108*x+81
(w + 2*z)^2 = 4*z^2+4*w*z+w^2
$
```

factor should behave similarly. Here is example output from **factor** when run with a single argument:

```
$ factor "x^4+12*x^3+54*x^2+108*x+81"
x^4+12*x^3+54*x^2+108*x+81 = (x + 3)^4
$
```

You should NOT write your own symbolic algebra program to do expansion and factorization. Use the **maxima** symbolic math program to do that for you. (You may need to install the **maxima**, **wxmaxima**, and **maxima-doc** packages.) Your programs should communicate with **maxima** using unnamed pipes. That is, they should set up one or more pipes, then fork a child process that hooks up the pipes to standard input and standard output and exec's the **maxima** program. The **maxima** program should only be exec'd once for each invocation of **expand** or **factor**.

Hints:

1. Play with the **wxmaxima** GUI interface first. It will display the proper **maxima** commands that you need to feed to the **maxima** command-line program. (You will want to change the 2d Display option to **none** in **wxmaxima** and set the **display2d** variable to **false** in **maxima** to get the output formatted as shown above.)

2. To determine how to properly format input for the **maxima** command-line program, try putting test input into a file and run **maxima** with redirected input and output:

   ```
   $ maxima -q < test_input.txt > test_output.txt
   ```

   (The '**-q**' option suppresses the **maxima** start-up message.) Examine the output file closely to determine the format of the **maxima** output.

3. Sometimes the **maxima** response consists of multiple writes to standard output and sometimes the response appears to be the result of a single write. The most fool-proof thing to do after sending a command to **maxima**, is to keep reading until you see the **maxima** prompt "**(%i#)**". You then know that **maxima** is ready for a new command.

4. Make sure that the **maxima** process terminates when the parent process terminates in all cases. (i.e., do not leave any zombie processes.) The command to terminate **maxima** is "**quit();**"

What to submit

- Provide a makefile named **Makefile** that will make all three programs for this assignment as the default target (typically called **all**). Each program must be a separate target.
- Create a tarfile or zipfile containing your (well-documented) program source files and makefile.
- Submit your archive using the submission system (http://submission.evansville.edu). The grading script only will make the project and check that executables named **expand** and **factor** are produced. It will not run anything.