**MCS 253P Lab 2**

1. Write a program to parseCmd shell commands into parts. Handle commands and arguments (words) and the operators <>&|. Remember words may contain paths like /bin/ls. Spaces are only required bertween words (which may be the command names and the arguments to the commands while they are optional between operators and words). Input will be read from stdin and will have one command per line and the output, sent to stdout, will be each token in the command, one per line. Use Bash I/O redirection to test this program on your saved command history like this:

$ history | parseCmd > BashTokens

You may set up a test file from your Bash prompt with this command. Look at the file with more or vim. Note history also adds history numbers. You can treat those as separate words, but I deleted them form my example below. Read [this](https://www.gnu.org/software/bash/manual/bashref.html#Simple-Commands) for more detail about bash commands.

$ history > testInput

Here is a sample input file

vi foo.cc>output&

cat<foo.cc>foo.output&

history|parseCmd>BashTokens

cat rolodex.c|tr A-Z a-z>output.foo&

Here is a what the output should be for just the last command above

cat

rolodex.c

|

tr

A-Z

a-z

>

output.foo

&

Note: do not even think about using strtok() to do this problem. It will only waste your time.

1. Write a function **atoi** (stands for “ascii” to “integer”)to convert a c-string of base-10 digits to a signed 32 bit decimal number and write a function **itoa** to convert a signed 32 bit decimal number into a c-string of digits (here is an example of use <http://fresh2refresh.com/c/c-type-casting/c-itoa-function/>). Write a main program, **testIntegers**, to read a series of lines as c-strings from standard input (using fgets()) then convert the string (e.g., stored in char s[100];) into a signed integer (e.g., stored in int value;) using your atoi function, then convert the integer back to a string using itoa then print the c-string to the standard output. If the generated string does not equal the input string (using **strcmp**()==0 for equality), print an appropriate error message to stderr. Use Bash I/O redirection to test this program on your sample input file of numbers like this:

$ testIntegers < numbers > myNumbers

Here is a sample of what numbers might contain ([32 bit integer](http://en.wikipedia.org/wiki/2147483647) -2147483647..2147483647)

12345

-12345

-1

1

0

2111111111

-2111111111

1. Write a program to convert English numbers into decimal and vice versa. E.g., one hundred fourty three thousand, one hundred, twenty five will convert to 143125. You may make the language regular as necessary, but it should be similar to the example above. You should handle any positive number under one million.

Here is a sample of what the input file might contain

one hundred fourty three thousand, one hundred, twenty five

five hundred twenty one thousand, three hundred fourty one