Name:
Partner:

General Problem Description

Write a program to parseCmd shell commands into parts. Handle commands and arguments (words) and the operators <>&|.

Additional Problem Specifics

1. Spaces are only required bertween words
2. 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.

Sample Input

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
&
```

Proposed Algorithm

Description:
Go through each location and act accordingly

Correctness:

Time Complexity:
O(n)

Space Complexity:
O(n)

C++ Implementation of Algorithm

```
#include <iostream>
#define STRMAX = 100

Void lineParse(string line){
        char single_element;
        for(single_element in line){
                switch(single_element){
                        case ' ': case'\t': case '\n':
                                cout << '\n';
                                break;
                        case '<':  case '>': case '&' case'|':
                                cout << '\n'<<single_element<<endl;
                                break;
                        default:
                                cout<<single_element;

                }

        }
}

Int main(){
        string line;
        cin.getline(line, STRMAX, '\n');
        lineParse(line);

}
```

Advantages/Disadvantages of Your Algorithm and Any Other Comments

Test Cases

```
 vi foo.cc>output&
expect:
vi
foo.cc
>
Output
&
```

```
 cat<foo.cc>foo.output&
expect:
cat
<
foo.cc
>
foo.output
&


 history|parseCmd>BashTokens
 cat rolodex.c|tr A-Z a-z>output.foo&
cat    < foo.cc >                          foo.output   &
/bin/ls
```

Screenshot of Compilation and Execution of Program Under Valgrind

Important Notes (you may erase these notes when pasting your screenshots):

- **Screenshots should demonstrate your program handling each of your proposed test cases. You may need to create your own custom inputs to demonstrate your test cases.**

- **Your program outputs should be CLEARLY LABELED on the terminal so that you (or any colleague that runs your program) can easily understand each of the following:**
  - **the details of the test case being tested**
  - **what your program's output is for that test case**
  - **whether your program correctly gave the desired output or not**
    - **one way is to provide the expected output to use as comparison**
    - **or you can use your own method, as long as the person running your program can easily understand from the output whether the program worked as expected or not**

  Since your screenshots capture your program output, this information should also be clearly visible within your screenshots!

- **To keep things simple, please address all your test cases within a single run of your program.**
  - **one way to do this by using functions to facilitate the testing of the different test cases**
  - **as a final test case, don't forget to include running your program with the test-case that is provided to you by the instructional staff**

- **Screenshots for each run of your program should start with a screen shot that includes your command to run the program under Valgrind and end with a screenshot that shows the final memory and error report.**

- **For additional notes, please see the Dr. Klefstad's HW submission guide.**