

Quinn Roemer

CISP - 440

Assignment 16

12/20/2018

Part 0 - Natural Language Generator

Description:

The goal of this assignment was to fix some code provided to us. To do this, we had to identify several issues in the code and remedy them. After we got the code working we were to modify the code to implement 4 additional grammars outlined in another document. Lastly, we were to create our own grammar and implement it.

Source Code:

```
//Code written by Professor Ross, Modified by Quinn Roemer
#include <iostream>
#include <string.h>
#include <time.h>

#pragma warning( disable : 4996)

using namespace std;

// Strings
#define SIZE 10000
char w[SIZE] = "A"; //our initial string of N's and T's
char buf[SIZE] = ""; //a buffer for string processing

//Productions
char P[][5][20] = {
    "Bc", "", "", "", "", // A -> Bc
    "Cb", "1", "", "", "", // B -> Cb|1
    "Ba", "1", "", "", "", // C -> Ba
};

// Terminal Expansions
char tTable[][20] = {
    "he said ", //a
    "she said ", //b
    "you lie.", //c
};

//Gets the cardinality for a row of the Productions table
int Cardinality(int row)
{
    int col = 0;

    // look for an empty string
    while (P[row][col][0])
        col++;
}
```

```

    return col;
}

//Returns true if there are Nonterminals in w
bool gotNs()
{
    for (int i = 0; w[i]; i++)
        if (w[i] >= 'A' && w[i] <= 'Z')
            return true;
    return false;
}

//Introduces verbosity
//Replaces terminals with big strings for humans to read and ponder
void expand()
{
    int i = 0;  buf[0] = 0;
    while (w[i])
    {
        int index = w[i] - 'a';
        strcat(buf, tTable[index]);
        i++;
    }

    // copy back to w
    strcpy(w, buf);
}

//Main function to execute.
void main(void)
{
    //Seed random number generator
    srand((unsigned)time(0));

    cout << "w = " << w << endl; //Print status

    //Process all of w repeatedly until all nonterminals are gone
    while (gotNs())
    {
        //Scan w and replace nonterminals with random productions
        int i = 0; int j = 0; buf[0] = 0; int card = 0;

        //For each w[i] in w
        while (w[i])
        {
            //Nonterminals
            if (w[i] < 96)
            {
                //Finding row index.
                j = w[i] - 65;

```

```

        //cout << "J: " << j << endl;

        //Scanning the row in productions for characters.
        for (int count = 0; count < 5; count++)
        {
            if (strlen(P[j][count]) > 0)
            {
                card++;
            }
            //cout << "Card: " << card << endl;
        }

        card = rand() % card;

        strcat(buf, P[j][card]);    //Copy production to buf
    }
    // Terminals
    else                            //Else w[i] is a terminal
    {
        char lilbuf[10] = { w[i], 0 }; //Put this terminal into a lil buffer
        strcat(buf, lilbuf);          //Copy terminal to buf
    }

    i++;                            //Next char in w
}

//Copy back to w
strcpy(w, buf);
cout << endl << "w = " << w << endl;    //Print status
}

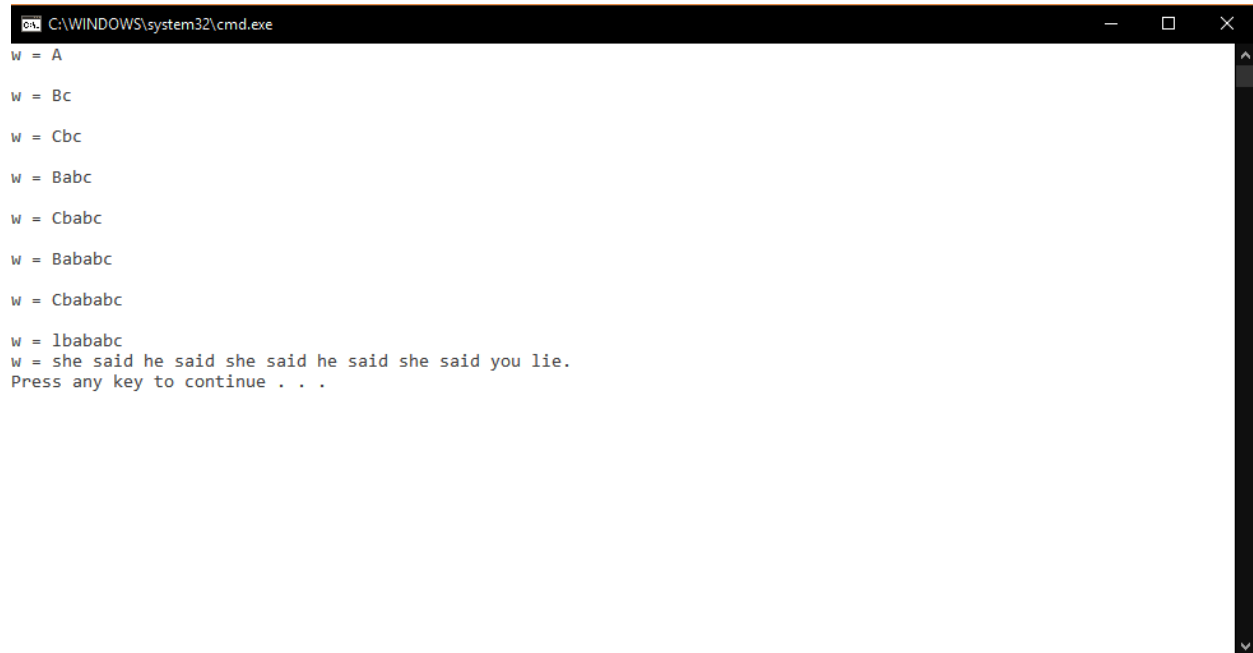
expand();
cout << "w = " << w << endl;
}

```

Output:

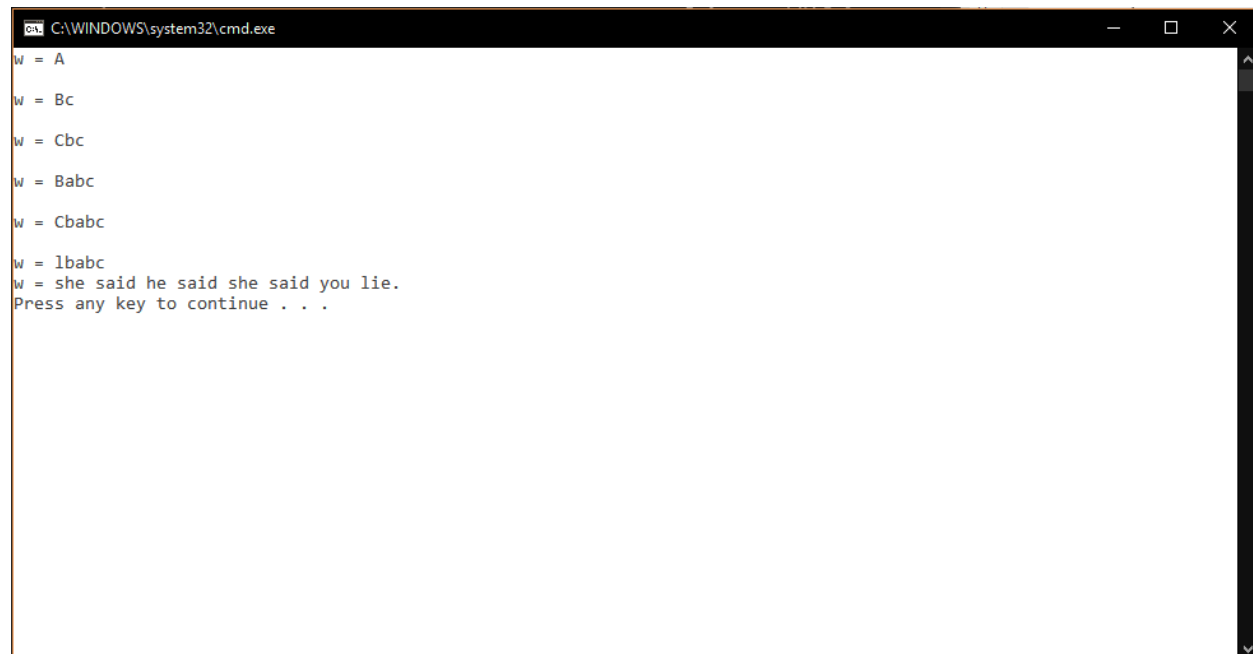
Note: The following 5 outputs will be for a given grammar. The first will be the default followed by the 3 outlined in the Natural Language PDF. Finally, the last output will be the CFG that I created myself. In addition, each output will contain two examples.

Default CFG



```
C:\WINDOWS\system32\cmd.exe
W = A
W = Bc
W = Cbc
W = Babc
W = Cbabc
W = Bababc
W = Cbababc
W = lbababc
W = she said he said she said he said she said you lie.
Press any key to continue . . .
```

Part 1 of 2



```
C:\WINDOWS\system32\cmd.exe
W = A
W = Bc
W = Cbc
W = Babc
W = Cbabc
W = lbabc
W = she said he said she said you lie.
Press any key to continue . . .
```

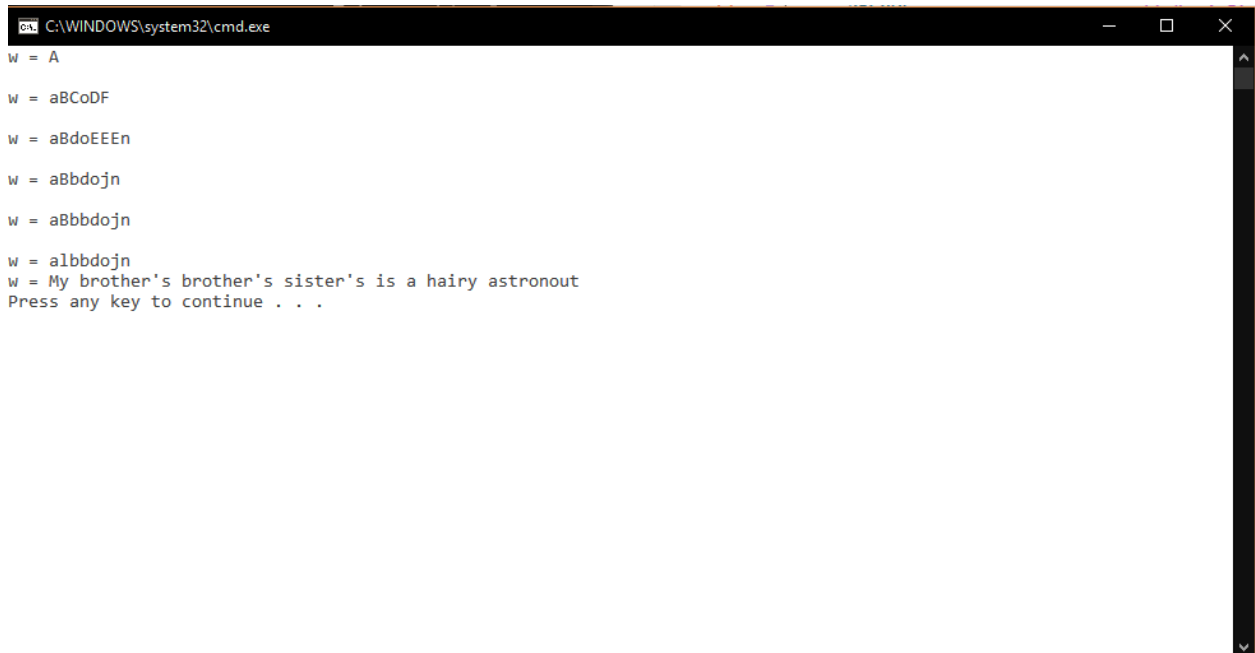
Part 2 of 2

Rockstar CFG



```
C:\WINDOWS\system32\cmd.exe
w = A
w = aBCoDF
w = aBbgoEEm
w = aBdbgoiim
w = aldbgoiim
w = My sister's brother's hairdresser is a excellent excellent noodlehead
Press any key to continue . . .
```

Part 1 of 2



```
C:\WINDOWS\system32\cmd.exe
w = A
w = aBCoDF
w = aBdoEEEn
w = aBbdojn
w = aBbbdojn
w = albbdojn
w = My brother's brother's sister's is a hairy astronout
Press any key to continue . . .
```

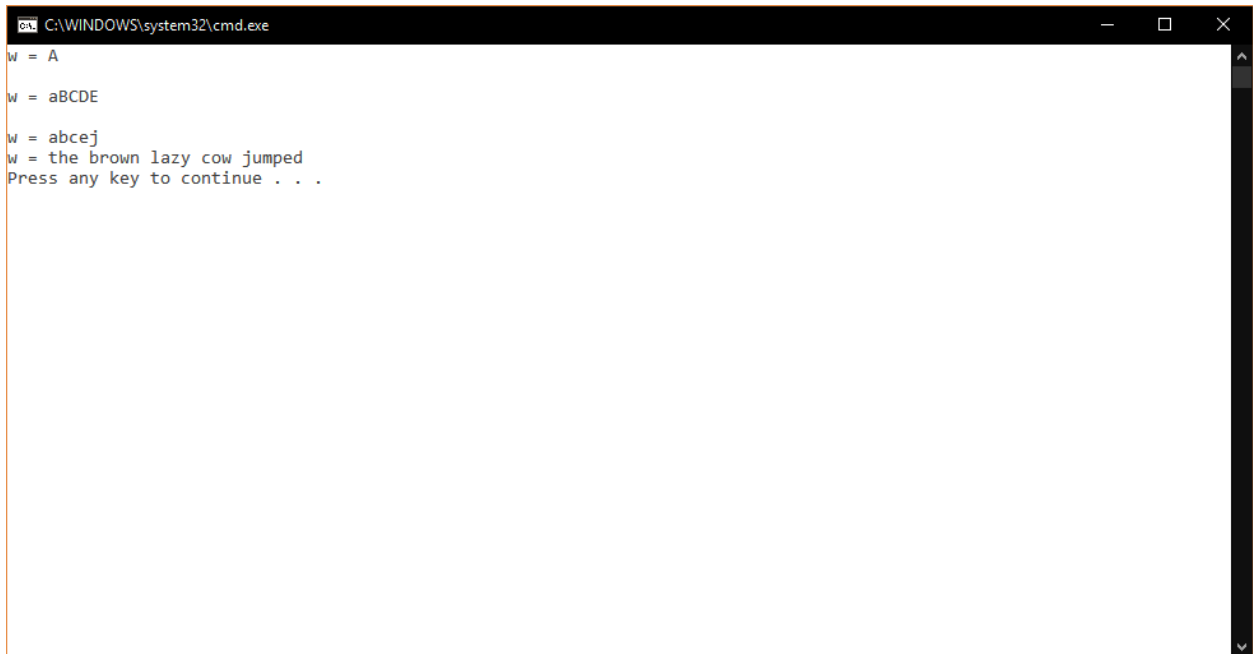
Part 2 of 2

Animal CFG



```
C:\WINDOWS\system32\cmd.exe
w = A
w = aBCDE
w = acdgi
w = the lazy fat liger swam
Press any key to continue . . .
```

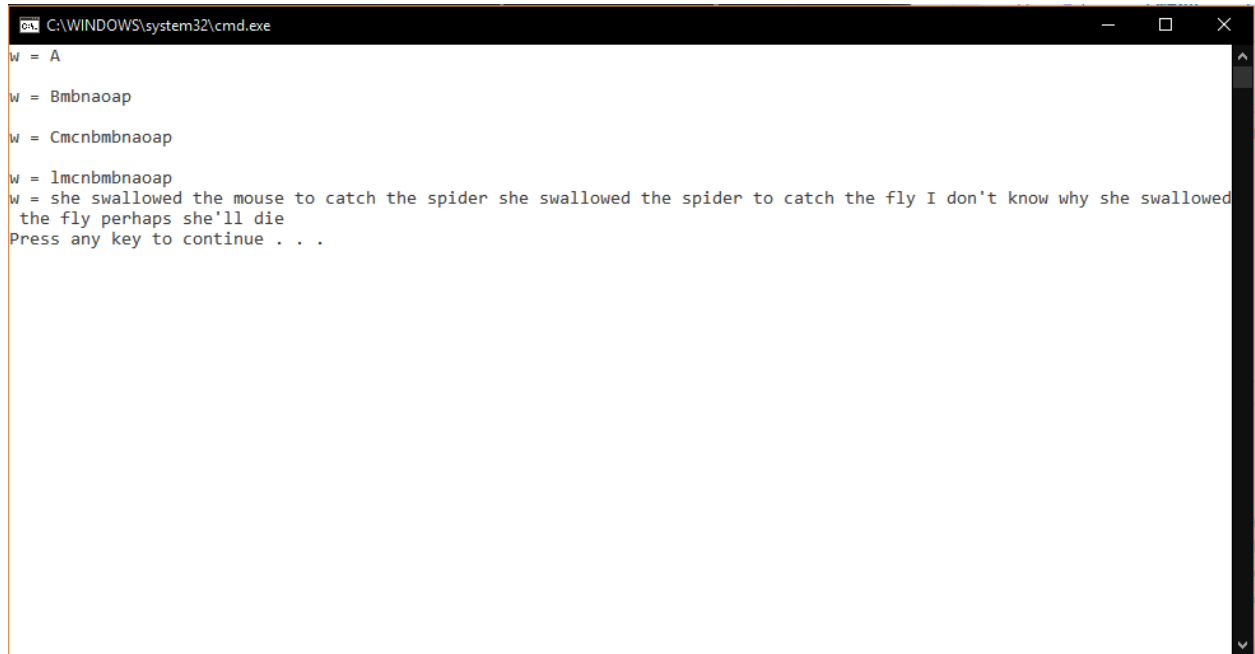
Part 1 of 2



```
C:\WINDOWS\system32\cmd.exe
w = A
w = aBCDE
w = abcej
w = the brown lazy cow jumped
Press any key to continue . . .
```

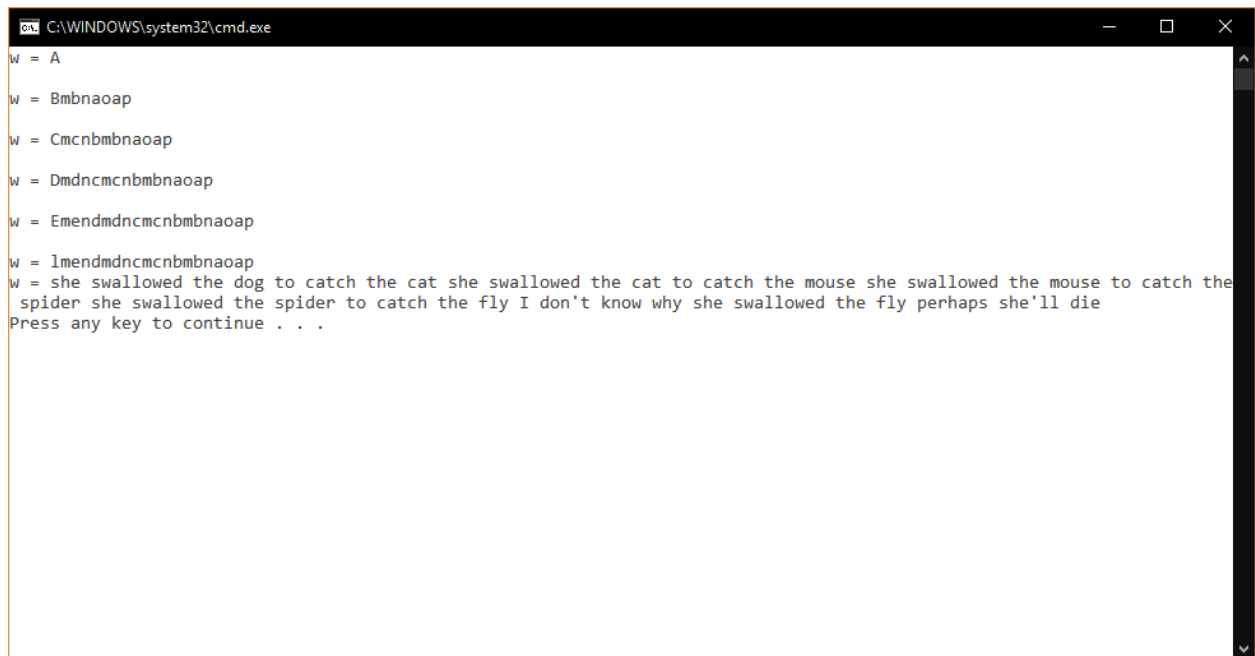
Part 2 of 2

She Swallowed CFG



```
C:\WINDOWS\system32\cmd.exe
w = A
w = Bmbnaoap
w = Cmcnbmbnaoap
w = lmcnbmbnaoap
w = she swallowed the mouse to catch the spider she swallowed the spider to catch the fly I don't know why she swallowed
the fly perhaps she'll die
Press any key to continue . . .
```

Part 1 of 2



```
C:\WINDOWS\system32\cmd.exe
w = A
w = Bmbnaoap
w = Cmcnbmbnaoap
w = Dmdncmcnbmbnaoap
w = Emendmdncmcnbmbnaoap
w = lmendmdncmcnbmbnaoap
w = she swallowed the dog to catch the cat she swallowed the cat to catch the mouse she swallowed the mouse to catch the
spider she swallowed the spider to catch the fly I don't know why she swallowed the fly perhaps she'll die
Press any key to continue . . .
```

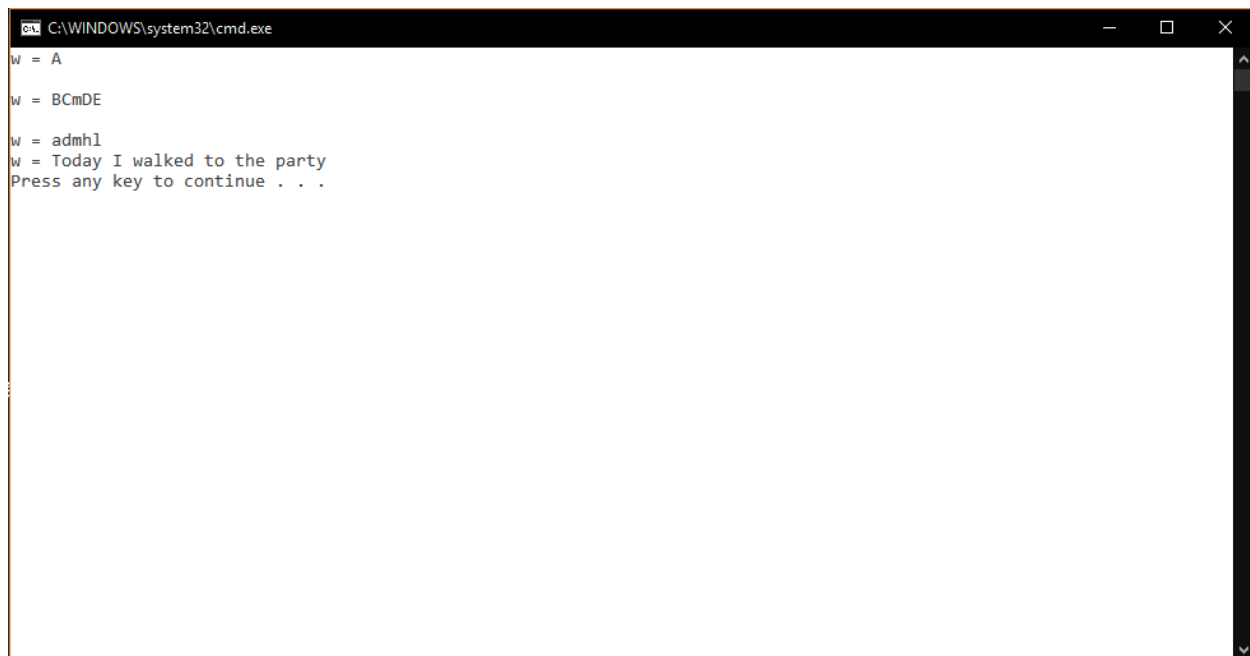
Part 2 of 2

CFG of my own design



```
C:\WINDOWS\system32\cmd.exe
W = A
W = BCmDE
W = bemgF
W = bemgk
W = Yesterday I flew to the store and died
Press any key to continue . . .
```

Part 1 of 2



```
C:\WINDOWS\system32\cmd.exe
W = A
W = BCmDE
W = admh1
W = Today I walked to the party
Press any key to continue . . .
```

Part 2 of 2

Conclusion

This last assignment was different from the rest of the semester. Instead of having to code something I had to fix something that already had been coded. Overall, I enjoyed the experience and was pleased when I got it working. As this is the last assignment of the semester I would like to thank you for teaching and wish you a Merry Christmas!