Our group went through the Word Cloud Program and used our “Good Code Primer” to enhance the code. We looked for ways to clean up, declutter, and increase readability of the program. We will go through and explain all of the changes we made to each function and show what the new code looks like after the changes.

Few notes before we get started: 1. We deleted some blank lines so that there was never more than one blank between blocks of code. 2. We deleted the line that had “import string” randomly in the middle of the program as it was useless. 3. We deleted the “demo()” function at the end of the program as we deemed it was useless and unnecessary.

1. **countWords(filename)**

```python
def countWords(filename):
    remove=',.?!()-!?"'
    wordcount = {}
    fin = open(filename,"r")
    for line in fin:
        for rem in remove:
            line=line.replace(rem,"")
        line=line.strip()
        words=line.split()
        if len(words)>0:
            for w in words:
                w=w.lower()
                if w in wordcount.keys():
                    wordcount[w]+=1
                else:
                    wordcount[w]=1
    fin.close()
    return wordcount
```

The only change we made in this function was removing the empty line between lines 17 and 18.

2. **deleteStopWords(wordcount)**

```python
def deleteStopWords(wordDictionary):
    fin = open("stopWords.txt","r")
    for line in fin:
        line=line.strip()
        if line in wordDictionary.keys():
            wordDictionary.pop(line)
        else:
            wordDictionary[line]=1
    fin.close()
    return wordDictionary
```
We made two changes in this function. We removed the comment “Remove stop words” as this was redundant based on the name of the function. The second change was we changed the use of “wordcount” to “wordDictionary” as worcount was already used in the first function. This change also makes clear the parameter must be a dictionary, and continues the use of camelback naming.

3. **makeTagCloud(filename)**

```python
def makeTagCloud(filename):
    words = countWords(filename)
    deleteStopWords(words)

    # Find the top 40 words
    lyst = []
    for key in words.keys():
        tup = [words[key], key]
        lyst.append(tup)
    lyst.sort()
    lyst.reverse()

    # clean this to an alphabetical list and mark the sizes needed
    end = min(40, len(lyst))
    highCount = lyst[0][0]
    lowCount = lyst[end-1][0]
    alpha = []
    print("Most common words were")
    print(lyst[:50])
    for x in range(0, end):
        alpha.append((lyst[x][1], lyst[x][0]))
    alpha.sort()

    body = ""
    for word, cnt in alpha:
        body = body + makeHTMLword(word, cnt, highCount, lowCount)
    box = makeHTMLbox(body)
    printHTMLfile(box, filename[:-4])
```

The first change we made in this function was to change the use of “wordcount” to be just “words” as we noticed that name had not been used. We decided to leave these comments and spaces for this function as they are useful in explaining what the blocks of this function are doing. The only comment change is we made line 13 a combination of two different comments.

After this function, we deleted several blank lines and the “import string” line.
4. **printHTMLfile(body,title)**

```python
def printHTMLfile(body,title):
    '''create a standard html page with titles, header etc.
and add the body (an html box) to that page. File created is title+'.html' '''
    fd = open(title+'.html','w')
    theStr="""""""
    <!DOCTYPE HTML PUBLIC "-/IETF//DTD HTML//EN">
    <html> <head>
    <title>""""+title+""""</title>
    </head>
    <body>
    <h1>""""+title+'<h1>'+""""\n'+body+'\n'+""""<hr>
    <address></address>
    </body> </html>
    """"
    fd.write(theStr)
    fd.close()
```

We did not change much to this function, just deleted a blank line between lines 9 & 10.

5. **makeHTMLbox(body)**

```python
def makeHTMLbox(body):
    # make an HTML box that has all the words in it
    boxStr = """"<div style="width: 800px;
background-color: rgb(250,250,250);
border: 1px grey solid;
text-align: center"">%s</div>
""""
    return boxStr % (body)
```

The only change we made with this function is making the multiline comment into a single line comment in line 2.
6. **makeHTMLword(word,count,high,low)**

```python
def makeHTMLword(word,count,high,low):
    ''' Font size is scaled between htmlBig and htmlLittle. 
    High and low represent the high and low counts in the document. ''
    htmlBig = 96
    htmlLittle = 14
    ratio = (count-low)/float(high-low)
    fontSize = htmlBig*ratio + (1-ratio)*htmlLittle
    wordStr = '<span style="font-size:%spx;">%s</span> ' % (str(fontSize), word)
    return wordStr % (str(fontSize), word)
```

The first change we made is we changed “cnt” to be “count” as a parameter. This makes it more clear what that input is for. We also shortened the multiline comment so that it is more straightforward and less filler. We also changed “fontsize” to “fontSize” for consistency with the camelback style.

As mentioned above, we deleted the “demo()” function as it seemed unnecessary and pointless. It also only tested the HTML functions, which you could test by calling the functions themselves rather than making a new function to test them.