1. **Timing: Part 1 (20 Points):**

   Compile and run the program without any extra optimizations, but with *profiling* for timing:

```
gcc -c -pg -O0 wordCounter.c
gcc -c -pg -O0 tree.c
gcc -c -pg -O0 list.c
gcc wordCounter.o tree.o list.o -pg -O0 -o wordCounterO0
```

   Run the program twice timing it both times, and answer the following:

```
a. $ ./wordCounterO0
b. File name to read: originOfSpecies.txt
c. Which algorithm would you like to run:
d. (1) Count words with tree
e. (2) Count words with linked-list
f.
g. Your choice? 1
```

buildTree() self seconds	**_____0.04___**

```
h. $ ./wordCounterO0
i. File name to read: originOfSpecies.txt
j. Which algorithm would you like to run:
k. (1) Count words with tree
l. (2) Count words with linked-list
m.
n. Your choice? 2
```

buildList() self seconds	**____0.38____**

2. **Timing: Part 2 (20 Points):**

   Compile and run the program *with* optimization, but with *profiling* for timing:

```
gcc -c -pg -O2 wordCounter.c
gcc -c -pg -O2 tree.c
gcc -c -pg -O2 list.c
gcc wordCounter.o tree.o list.o -pg -O2 -o wordCounterO2
```

   Run the program twice timing it both times, and answer the following:

```
a. $ ./wordCounterO2
b. File name to read: originOfSpecies.txt
c. Which algorithm would you like to run:
d. (1) Count words with tree
e. (2) Count words with linked-list
f.
g. Your choice? 1
```

buildTree() self seconds	**_______0.03___**

```
h. $ ./wordCounterO2
i. File name to read: originOfSpecies.txt
```

```
j. Which algorithm would you like to run:
k. (1) Count words with tree
l. (2) Count words with linked-list
m.
n. Your choice? 2
```

buildList() self seconds	**____0.23_____**

3. Parts of an executable (Points 20):

Please find the following inside of `wordCounterO0` by using `objdump` to show it (if it exists in the executable) or by using `objdump` to disassemble the code and showing where the code manipulates the heap or stack.

Show a *disassembly* or *objdump*. You do not have to show *all* of the objdump result if it is too long, but (1) please show the relevant output, and (2) please show the `objdump`command that you used to generate it.

a. The string `"File name to read: "` in `main()`
b. The local variable `rootPtr` in `buildTree()`
c. The code for `printList()`
d. The global variable `textLen`

Question	Command	Result
(A)	objdump -j .rodata -d wordCounterO0 _____	401248: 46 69 6c 65 20 6e 61 6d 65 20 74 6f 20 72 65 61 File name to rea 401258: 64 3a 20 00 72 00 43 6f 75 6c 64 20 6e 6f 74 20 d: .r.Could not _____
(B)	Since rootPtr is a local variable, it will be created in the Stack. Since there is no stack in the program it does not exist yet. _____	_____
(C)	objdump -j .text -d wordCounterO0 _____	00000000004010eb <printList>: 4010eb: 55 push %rbp 4010ec: 48 89 e5 mov %rsp,%rbp 4010ef: 48 83 ec 20 sub $0x20,%rsp

		4010f3: e8 28 f8 ff ff callq 400920 <mcount@plt> 4010f8: 48 89 7d e8 mov %rdi,-0x18(%rbp) 4010fc: 48 8b 45 e8 mov -0x18(%rbp),%rax 401100: 48 89 45 f8 mov %rax,-0x8(%rbp) 401104: eb 2c jmp 401132 <printList+0x47> 401106: 48 8b 45 f8 mov -0x8(%rbp),%rax 40110a: 8b 50 08 mov 0x8(%rax),%edx 40110d: 48 8b 45 f8 mov -0x8(%rbp),%rax 401111: 48 8b 00 mov (%rax),%rax _____
(D)	objdump -j .bss -d wordCounterO0	Disassembly of section .bss: 00000000006020c0 <stdin@@GLIBC_2.2.5>: ... 00000000006020c8 <stderr@@GLIBC_2.2.5>: ... 00000000006020d0 <called.4239>: 6020d0: 00 00 00 00 .... 00000000006020d4 <completed.6355>: 6020d4: 00 00 00 00 .... 00000000006020d8 <textLen>: ... _____

4. Compiler optimizations (Points 30):

Find and show at least 2 examples (total) of the following optimizations in either `wordCounterO0` or `wordCounterO2`.

A. usage of registers to hold vars (as opposed to the stack)
B. code motion
C. reduction in strength

For both:

- Tell if it exists in either `wordCounterO0`, `wordCounterO2`, or *both*, and

- Show these optimizations in the *disassembly* of the function that has it

Example 1:

Command: objdump -j .text -d wordCounterO2

This optimization is found where the local variables are stored in registers (Ex. r13) in the `wordCounterO2` assembly code under the "buildTree" method as follows:

```
0000000000400ce0 <buildTree>:
  400ce0:       55                      push   %rbp
  400ce1:       48 89 e5                mov    %rsp,%rbp
  400ce4:       41 57                   push   %r15
  400ce6:       41 56                   push   %r14
  400ce8:       41 55                   push   %r13
  400cea:       41 54                   push   %r12
  400cec:       53                      push   %rbx
  400ced:       48 81 ec 28 01 00 00    sub    $0x128,%rsp
  400cf4:       e8 47 fc ff ff          callq  400940 <mcount@plt>
  400cf9:       45 31 e4                xor    %r12d,%r12d
  400cfc:       48 89 bd b8 fe ff ff    mov    %rdi,-0x148(%rbp)
  400d03:       48 63 3d ce 13 20 00    movslq 0x2013ce(%rip),%rdi        # 6020
d8 <textLen>
  400d0a:       e8 21 fc ff ff          callq  400930 <malloc@plt>
  400d0f:       49 89 c5                mov    %rax,%r13
  400d12:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
  400d18:       48 8b 95 b8 fe ff ff    mov    -0x148(%rbp),%rdx
  400d1f:       8b 35 b3 13 20 00       mov    0x2013b3(%rip),%esi        # 6020
d8 <textLen>
  400d25:       4c 89 ef                mov    %r13,%rdi
  400d28:       e8 a3 fb ff ff          callq  4008d0 <fgets@plt>
  400d2d:       48 85 c0                test   %rax,%rax
  400d30:       0f 84 f3 00 00 00       je     400e29 <buildTree+0x149>
```

But this optimization is NOT found in the same section of code in `wordCounterO0`

`Command: objdump -j .text -d wordCounterO0`

```
0000000000400d35 <buildTree>:
  400d35:       55                      push   %rbp
  400d36:       48 89 e5                mov    %rsp,%rbp
  400d39:       48 81 ec 50 01 00 00    sub    $0x150,%rsp
  400d40:       e8 db fb ff ff          callq  400920 <mcount@plt>
  400d45:       48 89 bd b8 fe ff ff    mov    %rdi,-0x148(%rbp)
  400d4c:       48 c7 45 f8 00 00 00    movq   $0x0,-0x8(%rbp)
  400d53:       00
  400d54:       8b 05 7e 13 20 00       mov    0x20137e(%rip),%eax        # 6020
d8 <textLen>
  400d5a:       48 98                   cltq
  400d5c:       48 89 c7                mov    %rax,%rdi
  400d5f:       e8 ac fb ff ff          callq  400910 <malloc@plt>
```

```
400d64:       48 89 45 d8             mov    %rax,-0x28(%rbp)
400d68:       e9 30 01 00 00          jmpq   400e9d <buildTree+0x168>
```

Example 2:

Command: objdump -j .text -d wordCounterO2

This optimization is found where the local variables are stored in registers (Ex. rbx) in the `wordCounterO2` assembly code under the printList method as follows:

```
0000000000401030 <printList>:
 401030:    55                      push   %rbp
 401031:    48 89 e5                mov    %rsp,%rbp
 401034:    53                      push   %rbx
 401035:    48 83 ec 08             sub    $0x8,%rsp
 401039:    e8 02 f9 ff ff          callq  400940 <mcount@plt>
 40103e:    48 85 ff                test   %rdi,%rdi
 401041:    48 89 fb                mov    %rdi,%rbx
 401044:    74 25                   je     40106b <printList+0x3b>
 401046:    66 2e 0f 1f 84 00 00    nopw   %cs:0x0(%rax,%rax,1)
 40104d:    00 00 00
 401050:    8b 53 08                mov    0x8(%rbx),%edx
 401053:    48 8b 33                mov    (%rbx),%rsi
 401056:    31 c0                   xor    %eax,%eax
 401058:    bf 30 12 40 00          mov    $0x401230,%edi
 40105d:    e8 1e f8 ff ff          callq  400880 <printf@plt>
 401062:    48 8b 5b 10             mov    0x10(%rbx),%rbx
 401066:    48 85 db                test   %rbx,%rbx
 401069:    75 e5                   jne    401050 <printList+0x20>
 40106b:    48 83 c4 08             add    $0x8,%rsp
 40106f:    5b                      pop    %rbx
 401070:    5d                      pop    %rbp
```

This optimization is NOT found in the `wordCounterO0` assembly code:

Command: objdump -j .text -d wordCounterO0

```
00000000004010eb <printList>:
  4010eb:      55                    push   %rbp
  4010ec:      48 89 e5              mov    %rsp,%rbp
  4010ef:      48 83 ec 20           sub    $0x20,%rsp
  4010f3:      e8 28 f8 ff ff        callq  400920 <mcount@plt>
  4010f8:      48 89 7d e8           mov    %rdi,-0x18(%rbp)
  4010fc:      48 8b 45 e8           mov    -0x18(%rbp),%rax
  401100:      48 89 45 f8           mov    %rax,-0x8(%rbp)
  401104:      eb 2c                 jmp    401132 <printList+0x47>
  401106:      48 8b 45 f8           mov    -0x8(%rbp),%rax
  40110a:      8b 50 08              mov    0x8(%rax),%edx
  40110d:      48 8b 45 f8           mov    -0x8(%rbp),%rax
  401111:      48 8b 00              mov    (%rax),%rax
  401114:      48 89 c6              mov    %rax,%rsi
  401117:      bf f2 12 40 00        mov    $0x4012f2,%edi
  40111c:      b8 00 00 00 00        mov    $0x0,%eax
  401121:      e8 5a f7 ff ff        callq  400880 <printf@plt>
  401126:      48 8b 45 f8           mov    -0x8(%rbp),%rax
  40112a:      48 8b 40 10           mov    0x10(%rax),%rax
  40112e:      48 89 45 f8           mov    %rax,-0x8(%rbp)
  401132:      48 83 7d f8 00        cmpq   $0x0,-0x8(%rbp)
  401137:      75 cd                 jne    401106 <printList+0x1b>
```