Name: Fletcher O'Brien
Course: CSCI 312 Principles of Programming Languages
Assignment Deadline: April 2, 2025

Question 1

1. What is the bash command for generating a `ctags` file? **ctags -R**
2. What is the `vi` command for invoking generic keyword autocompletion? **Ctrl + N**
3. What is the `vi` command for invoking current buffer keyword autocompletion? **Ctrl + N**
4. What is the `vi` command for invoking whole line autocompletion? **Ctrl + X Ctrl + L**
5. What is the `vi` command for refining the word list as you type? **Ctrl + N**

Question 2

1. What is a definition in C? **A preprocessor directive to define macros. Example would be #define PI 3.14159**
2. What is a declaration in C? **Informs the compiler about the name of a variable or name, arguments, and return type of a function but does not allocate memory. Example would be int x; or int subtract(int a, int b);**

Question 3

What are three differences between arrays and pointers?

1. **Pointer variables can be assigned a value whereas array variables cannot. Example would be int a[7]; int *p; p = 1; a = 1 //this wouldn't work**
2. **Adding to a pointer is allowed but not to an array. Example would be int a[7]; int *p; p++; a++ //this wouldn't work**
3. **The sizeof method works with arrays but not pointers. Example would be Example would be int a[7]; int *p; sizeof(a); sizeof(p) //this wouldn't work**

Question 4 (Look at the Segments in an Executable)

Make a new directory called `Assignment3` in your ppl repo. Make a new directory called `Assignment3/Question4` that will contain your source code and executables for this question. Complete *Look at the Segments in an Executable* (Expert C Programming p. 142):

1. Implement 1 in a file called `1.c` with an executable called `1.out`. Record your answer to 1 here:
   ls -l 1.out: -rwxr-xr-x. 1 root root 24960 Apr 2 13:45 1.out
   size 1.out:
   text data bss dec hex filename
   1026 532 4 1562 61a 1.out

2. Implement 2 in a file called `2.c` with an executable called `2.out`. Record your answer to 2 here:
   ls -l 2.out:-rwxr-xr-x. 1 root root 24960 Apr 2 13:45 1.out
   size 2.out:
   text data bss dec hex filename
   1026 532 4032 5590 15d6 2.out

3. Implement 3 in a file called `3.c` with an executable called `3.out`. Record your answer to 3 here:
   ls -l 3.out:-rwxr-xr-x. 1 root root 29040 Apr 2 14:06 3.out
   size 3.out:
   text data bss dec hex filename
   1026 4560 8 5594 15da 3.out

4. Implement 4 in a file called `4.c` with an executable called `4.out`. Record your answer to 4 here:
   ls -l 4.out: -rwxr-xr-x. 1 root root 24968 Apr 2 14:09 4.out
   size 4.out:
   text data bss dec hex filename
   1066 532 4 1602 642 4.out
   local data not actually stored in executable, no big difference in size

5. Implement 5 in a file called `5.c` with an executable called `5d.out` for the debugging question and record your answer to the debugging question here:
   ls -l 5d.out:-rwxr-xr-x. 1 root root 30360 Apr 2 14:13 5d.out
   size 5d.out:
   text data bss dec hex filename
   1066 4560 8 5634 1602 5d.out
   When compiling for debugging the amount of data grows larger....

6. ...and an executable called `5o.out` for the optimization question and record your answer to the optimization question here:
   ls -l 5o.out:-rwxr-xr-x. 1 root root 29048 Apr 2 14:20 5o.out
   size 5o.out:
   text data bss dec hex filename
   1024 4560 8 5592 15d8 5o.out
   Noticably smaller than debug version

As you implement new requirements in your assignments for the rest of the semester, continue to use branching to get more practice.

Question 5 (Stack Hack)

Make a new directory called `Assignment3/Question5` that will contain your source code and executables for this question. Complete *Stack Hack* (Expert C Programming p. 146):

1. Compile and run the small test program (to discover the approximate location of the stack on your system) in a file called `stack_hack_1.c` with an executable called `stack_hack_1.out`. Record your answer here: **The stack top is near 0x7ffdada6ef4c**

2. Discover the data and text segment, and the heap within the data segment, in a file called `stack_hack_2.c` with an executable called `stack_hack_2.out`. Record your answer here: **data: 0x404028**
   text: 0x401156
   heap: 0x12462a0

3. Make the stack grow in a file called `stack_hack_3.c` with an executable called `stack_hack_3.out`. What's the address of the top of the stack now? Record your answer here: **The stack top is near 0x7ffd1f234b3c**
   The stack grew, new top at 0x7ffd1f234b38

Question 6 (The Stack Frame)

Make a new directory called `Assignment3/Question6` that will contain your source code and executables for this question. Complete *The Stack Frame* (Expert C Programming p. 151):

1. Manually trace the flow of control. Record your answer here: **main() to a(1) to a(0) to print "i has reached zero" to return to a(1) to return to main() to return**

2. Implement 2 in a file called `main.c` with an executable called `a.out`. Record your answer here:

   Breakpoint 1, 0x0000000000401158 in main () (gdb) next Single stepping until exit from function main, which has no line number information.

   Breakpoint 2, 0x000000000040112a in a () (gdb) next Single stepping until exit from function a, which has no line number information.

   Breakpoint 2, 0x000000000040112a in a () (gdb) next Single stepping until exit from function a, which has no line number information. i has reached zero 0x0000000000401162 in main () (gdb) backtrace 0 0x0000000000401162 in main () (gdb) next Single stepping until exit from function main, which has no line number information. 0x00007ffff7c3fee0 in $_{l}ibc_{s}tart_{c}all_{m}ain()from/lib64/libc.so.6Usinghostlibthread_{d}blibrary"/lib64/libthread_{d}b.so.1"$.

   Breakpoint 1, 0x0000000000401158 in main () (gdb) next Single stepping until exit from function main, which has no line number information.

   Breakpoint 2, 0x000000000040112a in a () (gdb) next Single stepping until exit from function a, which has no line number information.

Breakpoint 2, 0x000000000040112a in a () (gdb) next Single stepping until exit from function a, which has no line number information. i has reached zero 0x0000000000401162 in main () (gdb) backtrace 0 0x0000000000401162 in main () (gdb) next Single stepping until exit from function main, which has no line number information. 0x00007ffff7c3fee0 in $_{libc_start_call_main()from/lib64/libc.so.6}$