Reading. Read the following sections in the textbook.
- Ch 3: 3.7 to 3.8

Homework Questions. Do the following problems and turn in your answers as a single PDF document to blackboard. For each of the programming problems below, you must turn in your source code along with output showing your program compiles, runs, and produces the correct output. Note that this should be done on multiple tests for each program as appropriate. Your homework is due in blackboard at any time on the due date.

1. Do Practice Problem 3.32 from the textbook and write down your answers.

2. Do Practice Problem 3.34 from the textbook and write down your answers.

3. Do Practice Problem 3.35 from the textbook and write down your answers.

4. Rewrite your answer to Question 3 in HW6 to use a function call for converting the character to lower case. Your program should be a stand-alone assembly program. Your program must use the Linux / AMD64 ABI calling conventions. Name your function to_upper_case.

5. Rewrite your answer to Question 5 in HW6 to use a function for switching case (call your function switch_case. In addition, your switch_case function should call your to_upper_case function and should call a to_lower_case function (that you’ll need to write). Your program should be a stand-alone assembly program. Your program must use the Linux / AMD64 ABI calling conventions.

6. Write a C program that checks if a string consists of only letters. Your program should consist of a C driver file that calls a function contains_letter with the signature int contains_letter(char* str). The function should return 1 if str contains a letter and 0 if not. You must write the function using assembly (as shown in class). Note that a C string created in your driver program will end with a null terminator ('\0') symbol. Your driver should check different strings to ensure your function works correctly.

7. Write an assembly function that computes the product of an array of integers and a C driver program that tests your function. Name your function array_product. Your function should have the signature int array_product(int array[], int length), where length is the assumed length of the given array. Be sure to check different length arrays as part of your driver program.

8. Write an assembly function that pairwise adds two int arrays and a C driver program that tests your function. Name your function array_add. Your function should have the signature void array_add(int op1[], int op2[], int result[], int length). After calling the function, result[i] should hold the value op[i] + op2[i] for each i from 0 to length-1. Be sure to check different length arrays as part of your driver program.
9. Write an assembly function that implements the selection sort algorithm over an array of integers and a driver program that tests your function. Your function should have the signature `void selection_sort(int array[], int length)`. The result of calling your function should be that the values in `array` are sorted from smallest to largest value. Your driver program should test different input arrays to the function.

10. Write selection sort as a “regular” C function. Test it to make sure it works correctly (with your driver above). Use the `objdump -d` program to see the corresponding assembly code generated for the function. Compare the resulting function to your answer in Question 9 (note that you only need to consider the assembly in the section corresponding to the name of the function). Write down your observations and your explanations for the differences.