1. **Provide short answers** to each of the following questions:

   (a) What is stored in each *frame* on the *call stack*? Indicate, for each item, whether it is stored there by the *caller* or the *callee*.

   (b) What are *internal and external fragmentation*? Which does a *segregated fits* allocator exhibit?

2. **Write a function** in x86-64 assembly code that *reverses the elements in an array* of integers. The parameters of this function are:

   - *rdi*: The address at which the array of 64-bit integers begins.
   - *rsi*: The length of the array, given as a number of integer entries.

   The array should be reversed *in-place*—that is, the array to which *rdi* points should contain the result of the reversal. Here is a beginning to the function, which calculates, in *r10*, the address of the last entry in the array. **Complete this function.**

   ```assembly
   reverse_array:
   ;; Calculate the address of the final element.
   mov r9, rsi ; last_offset = length ...
   dec r9 ; last_offset = (length - 1) ...
   imul r9, 8 ; last_offset = (length - 1) * sizeof(int)
   mov r10, rdi ; last_addr = base ...
   add r10, r9 ; last_addr = base + offset
   ;; Perform the reversal...
   ```

3. Consider a garbage collected heap that occupies addresses 0x10000 through 0x90000. Furthermore, assume that this heap in managed by a *semi-space GC*. 
with the space divided at the halfway point, 0x50000. The lower address range is the \textit{from-space}, and the higher address range is the \textit{to-space}.

At the moment that the collector is triggered, there are a pair of objects in the \textit{from-space}, one at 0x10200, the other at 0x12300, each of which contains a pointer to the other. There is a single pointer to the first object from the root set.

**Describe the actions of the semi-space GC on these two objects.** That is, when the root pointer to 0x10200 is followed, detail how the GC will act on these two objects, leaving them in the \textit{to-space} and correctly pointing to one another. Choose addresses within the \textit{to-space} to which the objects move, and show how the pointers are correctly updated.

4. Recall the four memory regions of a process: \textit{text/code}; \textit{statics}; \textit{heap}; \textit{stack}.

Answer the following questions about these regions:

(a) Where does each appear in the virtual address space?
(b) What is stored in each?
(c) How is each created and managed? That is, what part(s) of the program and/or the system are responsible for each region?