Unit 2 Problem 1: C & Assembly

Answer the questions below about the following x86-64 assembly function:

```
mystery:
    movl  $0, %edx              #Line 1
.L2:
    cmpl %esi, %edx             #Line 2
    jge .L4                    #Line 3
    movslq %edx, %r8           #Line 4
    leaq (%rdi,%r8,2), %rcx    #Line 5
    movzwl (%rcx), %r8d        #Line 6
    leal (%r8,%r8,2), %r8d     #Line 7
    movw %r8w, (%rcx)          #Line 8
    addl $3, %edx              #Line 9
    jmp .L2                    #Line 10
.L4:  retq                    #Line 11
```

(A) What C variable type would %rdi hold in the corresponding C program?

(B) What C variable type would the 2nd argument be in the corresponding C program?

(C) Fill in the missing C code that is equivalent to the x86-64 assembly above:

```
    mystery(__________, __________) {
        __________ edx = __________;
    }
```

(D) Briefly describe what this function is doing at a high level.
Unit 2 Problem 2: The Stack & Procedures

The recursive function `all_even()` returns 1 (true) if every element in an `int` array is even and 0 (false) otherwise. Its x86-64 disassembly is shown below:

```c
int all_even(int* arr, int len) {
    if (!len) {
        return 1;
    } else {
        return !(*arr & 0x1) && all_even(arr + 1, len - 1);
    }
}
```

0000000000401126 <all_even>:

0000000000401126:  85 f6  test %esi, %esi
0000000000401128:  74 29  je  401153 <all_even+0x2d>
000000000040112a:  f6 07 01  testb $0x1, (%rdi)
000000000040112c:  74 06  je  401135 <all_even+0xf>
000000000040112e:  b8 00 00 00 00  mov $0x0, %eax
0000000000401134:  c3  retq
0000000000401135:  48 83 ec 08  sub $0x8, %rsp
0000000000401139:  83 ee 01  sub $0x1, %esi
000000000040113c:  48 83 c7 04  add $0x4, %rdi
0000000000401140:  e8 e1 ff ff ff  callq  401126 <all_even>
0000000000401145:  85 c0  test %eax, %eax
0000000000401147:  74 05  je  40114e <all_even+0x28>
0000000000401149:  b8 01 00 00 00  mov $0x1, %eax
000000000040114e:  48 83 c4 08  add $0x8, %rsp
0000000000401152:  c3  retq
0000000000401153:  b8 01 00 00 00  mov $0x1, %eax
0000000000401158:  c3  retq

(A) What is the return address to `all_even` that gets stores on the stack? Answer in hex.

(B) Assume main calls `all_even(arr, 4)` with `int arr[] = {2, 4, 6, 8}`. Fill in the stack layout diagram below as this call to `all_even` returns to main. For unknown values, write “unknown”.

(C) Notice that there are no push instructions in this code, yet we are modifying our argument registers on each recursive call. Why did the compiler choose to not save these values?