1a) If memory is laid out in this manner, what happens when the stack and the heap meet in the middle? Explain, in your own words, what would occur and what the consequences of that event would be on the program.
**Answer**: If a program tries to allocate memory on stack, it will encounter the "stack overflow" error and the program will crash. If a program tries to call malloc to allocate memory on heap, the operation will fail and return a NULL pointer because there is not enough memory to allocate.

1b) Describe a reason for using such a table, *i.e.*, a case in which a program calling operating system addresses directly might fail in some cases, while one using the system call table would not.
**Answer**: System call table provides a fixed interface to programs, regardless of changes to the OS, because the table is always located at a specific address (e.g., 0x1000) so the program can reliably access it from the same location when OS is upgraded.
For example, if a program directly calls 0x1008 to load data into memory. After an OS update, that function is moved to 0x1508. The program calls 0x1008 will now crash.
Nevertheless, if the program calls the system call table entry 0x05, and the table maps 0x05 to the new address 0x1508 after the OS is upgraded, the program continues to work seamlessly.

1c)What is wrong with the following C code fragment? Describe the issue or issues and how you would need to modify to code to avoid any potential problems.

```
char* block = calloc(2048);

memset(&block, 0xFF, 2048);
```

**Answer**:
1: calloc function requires two arguments. The first argument specifies the number of elements, and the second argument specifies the size of each element.
2: passing the address of the pointer "block", this will cause block pointer no longer points to the memory block of the allocated place.

**Correct code**:

```
char* block = calloc(2048, sizeof(char));

memset(block, 0xFF, 2048);
```