1. a) Yes, they are part of system call's Application Programming Interface, and they are the only way to interact between computer program and OS kernel.

   ## Notes

   - **System Calls**
     - Is issued by a client
     - Is the only entry points into the kernel system
     - Provides services via API or Application Program Interface
     - Has five different types of calls

Types of System Calls	Windows	Linux
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()

   ## Example

   `open()`, `read()`, `write()`, `close()`, `mkdir()` are other examples of system calls

   ## References

   1) Tutorials Point, Types of System Calls, link

   b) Malloc allocates heap memory, free deallocates heap memory, and it is user's responsibility to manage heap memory.

   ## Notes

   - **Memory API**
     - Has two types of memory
       1. **Stack**
          * Is also called **automatic memory**
          * Allocations and deallocations are managed by compiler

           ∗ Deallocates memory by the end of function call

     2. **Heap**

           ∗ Is long-lived

           ∗ Allocation and deallocation are managed by user

           ∗ Creates **memory leak** if memory not freed

           ∗ **valgrind** is a useful heap memoery debugging tool link

– `malloc()`

     ∗ Is a C library call

     ∗ **Syntax:** `void *malloc(size_t size)`

     ∗ Allocates a block of `size` bytes to **heap memory** and if successful, returns a pointer to it

     ∗ Returns `NULL` if memory allocation is unsuccessful

Example

```
int *x = malloc(10 * sizeof(int));
```

– `free()`

     ∗ Is a C library call

     ∗ Frees heap memory that is no longer in use

Example

```
int *x = malloc(10 * sizeof(int));
...
free(x);
```

– `brk()`, `sbrk()`, `mmap()`

     ∗ Are system calls for memory management

- **Buffer overflow**
  - is an error that occurs when not enough heap memory is allocated

Missing + 1

```
char *src = "hello";
char *dst = (char *) malloc(strlen(src)); // too small!
strcpy(dst, src); // work properly
```