Question 1:
copyin() takes a block of memory from the user-level program which is defined by the user-level address and the length LEN, and copy it to the kernel address DEST.
copyout() Copy a block of memory of length LEN from kernel address SRC to user-level address USERDEST.

Question 2:
They both specify the type of pointer. UIO_USERSPACE is used when the loaded segment is not executable, while UIO_USERISPACE is used for an executable loaded segment. If we want to copy within the kernel space we use UIO_SYSSPACE.

Question 3:
After going back to user mode, vnode is not used anymore. Therefore, vnode can be destroyed by calling vfs_close, and the file is released so other processes can use it. Also, every time when an error occurs, vfs_close is called to destroy the vnode.

Question 4:
md_usermode(0, NULL, stackptr, entrypoint);

Question 5:
It indicates that the type of the pointer which is used for the user address space.
it is a one-byte structure that points to a user level address space.
it is a pointer to a one-byte structure in order to be separated from other pointers.

Question 6:
Right now, kill_curthread() calls panic and does not handle any type of user-level exceptions and that is what we are going to implement.

Question 7:
In both functions Interrupts are enabled.

Question 8:
vfs_open

Question 9:
vop_open, vop_close, vop_reclaim, vop_read, vop_readlink, vop_getdirentry, vop_write, vop_ioctl, vop_stat, vop_gettype, vop_tryseek, vop_fsync, vop_truncate, vop_namefile, vop_creat, vop_symlink, vop_mkdir, vop_link, vop_remove, vop_rmdir, vop_rename, vop_lookup, vop_lookparent
We don’t need to create two vnodes since vop_creat has a flag EXCL. If the file already exists, it can use vop_lookup to hand back the vnode.