- **ifree Algorithm:**

```
1   ifree(inode_no){
2       Increment free inode count;
3           If (super block locked) return;
4           If (superblock inode list full) {
5                   if (inode number <remembered inode)   Set remembered inode as input
6                   inode ;
7            }
8             else
9                   Store inode number in inode list;
10      return;
```

Explain the need for the condition in line number 6: "(inode number <remembered inode) "

- Let us consider a storage device with a UNIX file system. The inode number for the root directory is 2. Every file in the storage device has unique names. Let us consider that

  "/ict/ufg/nvidia" is a valid path for a file named "grok"

  And let us also consider that the entries in a directory are:

.	is inode 34346
..	is inode 987
mr.ed	is inode 10674
joe.txt	is inode 8767
nvidia	is inode 67871
otherdir	is inode 2345

  What are the different values of the inodes for the directory ict and ufg?

- Why do we need to keep a complete list for Free Data Blocks, but only an incomplete Free Inode List?

- Incore Inodes have some extra fields than the Disk Inodes. Some of those fields are: **Locked, Changed, Inode Number, Pointer Fields, Reference Count.** Explain why do we need these **five (5)** fields in Incore Inodes and not in Disk Inodes.

- With the help of a diagram(s) explain how Free Data Block list is maintained, how it is updated when blocks are allocated and also freed.

- What is the advantage of allocating consecutive blocks to a file? Why is it not always possible to do so in UNIX file system.

- What are Incore Inodes and why do we need them?

- If block size is 512 K bytes and the address of a block takes 8 bytes then what is the size of the largest possible file in the corresponding UNIX file system? Show your calculation. **(WARNING : DO NOT MEMORIZE THE NUMBERS)**

```
Algorithm iget
input: file system inode number
output: locked inode
{
while (not done){
        if (inode in inode cache){
                        if (inode locked){
                        sleep (event inode becomes unlocked);
                        continue;  /*loop back to while */
                        }
                /*special processing for mount points */
                if (inode on inode free list)
                remove from free list;
                increment inode reference count;
                return (inode);
        }
        /*inode not in inode cache */
        if (no inodes on free list) return (error);
        remove new inode from free list;
        reset inode number and file system;
        remove inode from old hash queue, place on new one;
        read inode from disk (algorithm bread);
        initialise inode (e.g. reference count to 1);
        return (inode);
}
```

- When an inode is found form the free list why do we need to remove it from the old hash queue and place it onto a new one?
- Why is error returned when a free inode is not found instead of waiting for an inode to be free?
- We observe that there is a hash queue and a free list. Can an inode both be in the hash queue and the free list at the same time? Explain your answer.