Problem 1:
A file system uses UNIX inode data structure which contains **8** direct block addresses, **2** single indirect blocks, **2** double indirect blocks and **2** triple indirect blocks. The size of each block is **32 Bytes** and the size of each block address is **4 Bytes**. Find the maximum possible file size?

Solution:
Total # of pointers in one data block = 32/4 =8
Total # of pointers = 8 + (2*8) + (2*8*8) + (2*8*8*8) = 1176
Maximum file size = 1176 * 32 = **37632 Bytes** = **36.75 KB**

Problem 2:
A file system has inode size = 512B and block size = 4KB. First 3 blocks contain superblock, data bitmap and inode bitmap. Now calculate the address of the inode number 23. The disk is sector addressable and the size of each sector is 256 B. Find out the sector address of the inode block.

Solution:
Inode start address = 3*4 KB = 12 KB
Offset = 23 * 512 B = 11.5 KB ≈ 11 KB
Address of inode 23 = 12 + 11 KB = 23 KB
Block number = 23*512 B / 4 KB = 11.5 KB / 4 KB = 2.875 ≈ 2
Sector address = {(2*4 KB) + 12 KB} / 256 B = 80

Problem 3:

An existing file named “a1” needs to be read which is allocated in **2** data blocks.

- Path of the file: “/new/one/a1”
- To read the file it was opened first by open() system call.
- After opening the file read() system call was issued in the file to read the contents.

Illustrate the file access path timeline according to the scenario described above.

Solution:

	data bitmap	inode bitmap	root inode	new inode	one inode	a1 inode	root data	new data	one data	a1 data [0]	a1 data [1]
			1. read								
							2. read				
				3. read							
open(/new/one/a1)								4. read			
					5. read						
									6. read		
						7. read					
						8. read					
read()										9. read	
						10. write					
						11. read					
read()											12. read
						13. write					

Problem 4:

A file named “b1.c” has been created by create() system call.

- Path of the newly created file: “/abc/def/b1.c”
- After creating the file write() system call was issued in the file to write new contents and after the write operation the file has been allocated in **4** data blocks.

Illustrate the file access path timeline according to the scenario described above.

Solution:

	data bitmap	inode bitmap	root inode	abc inode	def inode	b1.c inode	root data	abc data	def data	b1.c data [0]	b1.c data [1]	b1.c data [2]	b1.c data [3]
create(/abc/def/b1.c)			1. read										
							2. read						
				3. read									
								4. read					
					5. read								
									6. read				
		7. read											
		8. write											
									9. write				
						10. read							
						11. write							
					12. write								
write()						13. read							
	14. read												
	15. write									16. write			
						17. write							
write()						18. read							
	19. read												
	20. write												
											21. write		
						22. write							
write()						23. read							
	24. read												
	25. write												
												26. write	
						27. write							
write()						28. read							
	29. read												
	30. write												
													31. write
						32. write							