Team Assignment

This assignment is to simulate the disk allocation and free-space management implementation of a file system. The objective is to evaluate the various methods to determine its efficiency (i.e. speed in accessing file content) and flexibility (i.e. ability in keeping different file sizes). You are to form a team of 5/6 students and implement the following with a program. The program must be written in C.

Your hard-disk is an integer array of size 500 – int hard_disk[500];

Below are what you need to implement:

- Directory structure: You need a directory structure to keep track of the basic file information and the location of where the file content is stored. You can either use the linear or the hash table method.
- Disk blocks allocation: The program needs to implement all three disk blocks allocation methods (i.e. Contiguous, Linked and Indexed).
- Free-space management: You can use either bit-map or linked list methods to determine the free blocks locations.

The physical store is represented by an array of 500 integer entries with 5 entries as a disk block.

Directory structure: The first two blocks are allocated for directory structure, where the information of all the files in that directory is stored.

Filenames: Filenames are integers and the accepted filenames are from 0 to127.

File contents: File contents accepted in this hard-disk are 4 byte positive integer values.

An example file and its contents:

Filename: 111

Contents: 32, 678, 91, 100

Table 1, as shown below, is a sample instance of the first 5 blocks of the physical store containing two different files stored in linked method.

File 1: 121 -> 13, 14, 17, 18, 19, 23

File 2: 27 -> 201, 202, 203

Index	Block	File Data
0	0	121, 2, 4
1	0	27, 5, 5
2	0	
3	0	
4	0	
5	1	
6	1	
7	1	
8	1	
9	1	
10	2	13
11	2	14
12	2	17
13	2	18
14	2	4

Index	Block	File Data
15	3	
16	3	
17	3	
18	3	
19	3	
20	4	19
21	4	23
22	4	-1
23	4	
24	4	
25	5	201
26	5	202
27	5	203
28	5	-1
29	5	

Table 1

Each entry in the array holds one number. The number can be either the file content or a pointer representing the block number of another block. This will apply to all the array entries except the directory structure which can contain multiple attributes as shown in the above example. The multiple contents are stored in a single integer (the first byte of the integer representing filename, second byte representing start block. Third byte representing end block, etc). This structure will vary according to the allocation method. But each entry in the directory structure can only contain the information for one file.

Below are the key steps of what the program should do:

1. Your program will read a csv file as the input in the following format. The first field is the file operation which includes add, read or delete a file. The second filed is the file identifier and the remaining are the data of the file. Each line represents an operation the program needs to perform to a single file.

   add, 121, 101, 12, 10, 14, 105, 106
   add, 27, 201, 202, 203
   read, 121
   delete, 27

2. Based on a disk block allocation method, the program will complete the operation as stated and output the status of the action. For add function it says if the file is successfully added or not. For read function the file should be read from the memory and the contents should be printed out. For delete function the file should be deleted and the status should be printed out.

3. The final output from the program should be a disk map showing the full contents of the physical store. You can use table 1 as a guide. It should also show the total number of added files, the overall size (i.e. number of integer). This should also apply to the read operation.

4. Step 1 to 4 should be in a loop so that more operations can be performed to see how the disk image changes.

Pick another disk allocation method and repeat the program. All 3 methods should be simulated – in 3 different programs.