CSI 2131 A/B
Winter 1999 Final Exam
Professor: Ken Barker

Wednesday, April 21, 9:30

Family Name		
Given Name		
Student Number		
Section (circle one)	A	B

Notes:
1. This is a closed book exam. Textbooks, notes, cheat sheets and little papers rolled up inside your ball point pen are **not** allowed.
2. Calculators, computers, abaci and all other computing devices are **not** allowed.
3. There are 12 pages. Write your name and student number on **every** page.
4. There are 6 questions. Answer **all** 6 of them.
5. Write **all** answers and work in the space provided. Use **no** other paper.
6. You have 3 hours to complete the exam.

Marks:

Question						
1 (13 marks)	2 (10 marks)	3 (7 marks)	4 (7 marks)	5 (8 marks)	6 (5 marks)	*Total* *(50 marks)*

Question 1: Cosequential Processing and Indexing

A ski club keeps a file `members.info` for its members. Here is the information that is kept in the file for each member:

```
Member Code
Family Name
Given Name
Address
Postal Code
Telephone Number
Skill (beginner, intermediate, expert)
```

The *member code* is the primary key and is a unique field consisting of the first three letters of the *family name*, the first three letters of the *given name* and a random 3-digit number. The file `members.info` is sorted on the primary key.

a) Every day, some members quit the club and some members join the club. The *member codes* for the quitters are kept in a separate file called `quitters.today`, which is sorted on *member code*. The record information for the joiners is kept in a separate file called `joiners.today`. The `joiners.today` file has the exact same format as `members.info` and is also sorted on *member code*.

   Write an algorithm that processes all three files cosequentially, writing an updated version of `members.info`. The updated version of `members.info` should still be sorted on *member code* and should contain the new records from `joiners.today` and all of the original records from `members.info` *except* those whose primary key appears in `quitters.today`.

   If there is a key in `quitters.today` that does not appear in the original `members.info`, display an error and continue. If there is a record in `joiners.today` that has a key that already exists in `members.info`, display an error and continue.

<more space next page>

b) The ski club also keeps a secondary index file called `postal_codes.ind`. The file contains one record for each member. Each record contains the first three characters of a member's postal code and the RRN of the member's record in `members.info`. The file `postal_codes.ind` is sorted on the three-character postal code.

Write an algorithm that takes the first three characters of a postal code (call them `code2find`) and writes out the names (*family name*, *given name*) of every member who has a postal code starting with `code2find`. Your algorithm must use the secondary index file `postal_codes.ind`.

Question 2: Hashing

My copy of the online Collins English dictionary has 311,587 words in it. The shortest words have 1 letter. The longest word has 30 letters. The dictionary is currently sorted with each dictionary word as the primary key. I would like to have the dictionary entries in a hash table instead of a sorted file.

a) Give a possible hashing function for mapping words in the dictionary to hash table positions. You may assume any table size you like (500,000 entries wouldn't be bad, with entries numbered from 0 to 499,999). You may also assume that words consist of *only* the lower case alphabetic characters a-z.

b) Now assume the hash table has been created using your hashing function from part a). Hash table entries are buckets containing 5 records each; each record is 100 bytes. Simple *linear probing* (or *progressive overflow*) was the collision resolution technique in building the hash table. Write an algorithm that takes a word as a string (call it `word2find`) and finds its record in the hash table. You may assume that your hashing function from part a) has already been implemented and you can simply call it as a function call (`h(word)`).

Question 3: Compression

Decode the following LZW-encoded sequence. You may assume that entries 0-255 in the dictionary already contain the 256 ASCII characters. For example, the dictionary entry for code `<100>` is the letter 'd'. You must show the decoded string and the resulting dictionary entries (from 256-268).

```
<100><111><95><256><103><95><116><104><101><258><111><260><256><111>
```

95	_
...	…
100	d
101	e
102	f
103	g
104	h
...	…
111	o
112	p
113	q
114	r
115	s
116	t
...	…
255	□
256	
257	
258	
259	
260	
261	
262	
263	
264	
265	
266	
267	
268	

Question 4: Sorting

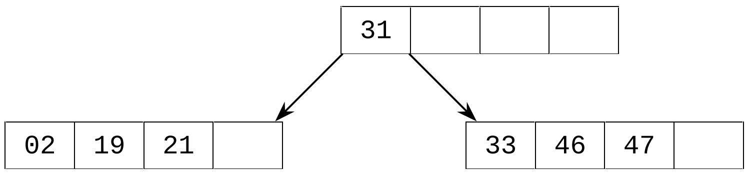
Show the steps of a heapsort on the following keys using replacement selection. Your heap can hold only 5 keys at a time. The result will be one or more sorted runs. Show the output runs, marking clearly where one run ends and the next begins.

```
Dam Ron Igo Lan Wad Sam Pat Jas Mag Ted Sha Rad Bru Ste Vac Mar
```

Note: you do not necessarily have to build an upside-down heap under the active heap; if you don't, you will just have to rebuild the heap once the active heap is empty before continuing to sort.

Question 5: B-Trees and B* Trees

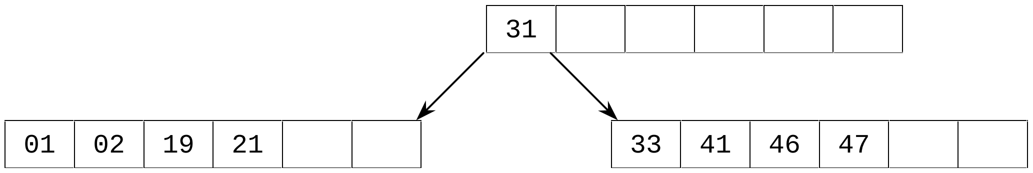
a) Here is a B-tree of order 5.



Add the following keys in order to the B-tree.

```
01 23 13 41 55 85 25 90 14 65 10
```

b) Here is a B* tree of order 7.



Add the following keys in order to the B* tree.

`23,13,14,55,85,25,90,65,80`

Question 6: Storage Reclamation

Here is a data file containing variable length records. Fields are separated by '`,`' and records are separated by '`;`'.

```
Harris,Lawren,North Shore,Lake Superior,1926;Krieghoff,Cornelius,
Owl's Head and Skinner's Cove,Lake Memphremagog,1859;Fortin,Marc-
Aurèle,Landscape,Hochelaga,1931;Lismer,Arthur,A September Gale,Ge
orgian Bay,1921;Carr,Emily,Indian Hut,Queen Charlotte Islands,193
0;Légaré,Joseph,Cholera Plague,Québec,1832;
```

Perform the following record deletions and additions *in order*. Use the reclamation policy (First-Fit, Best-Fit or Worst-Fit) that is the *most* space efficient (wastes the least space) for this particular file and these particular deletions and additions. Show the resulting file and state which policy was most space efficient.

i) Delete the `Lismer` record
ii) Delete the `Krieghoff` record
iii) Add the record `Morrice,James Wilson,Café el Pasaje,Havana,1919;`
iv) Add the record `Brooker,Bertram,Alleluiah,1929;`
v) Add the record `Clark,Paraskeva,Petroushka,1937;`