

## Implementation of Heap File Organization

Read Heap File organization from Raghu Ramakrishnan's book.

We will consider the heap file as a doubly linked list of pages.

Each page will have a fixed size and it will be given as an input parameter.

Each page is divided into two parts: Data area and Book keeping area

The data area grows from beginning of the page to the end of the page. The book keeping area grows in the reverse direction (from end to the beginning). A new blank page has four integers stored at the end of page: address of previous page, address of next page, count of the number of records in the page, and start address of free space within the page. For a new blank page, count of records is zero and the free space begins at the address zero within the page.

Assume that each integer requires 4 bytes of storage.

To store a record into the page, you need to create a directory entry in the book keeping area. This entry will store pointer to the beginning of the record in the page. Assume that this pointer is an integer value. We do not store pointer to the end of record explicitly. To store a record of size  $R$ , the space occupied in the page is  $R+4$  bytes. The records are stored in the page without leaving any empty space in between.

Your heap file has to support three operations:

Insert

Search

Status

Insert:

Parameter: record size, value of primary key

Assume both these values to be integers.

Assume that there are no duplicates.

You should insert the record in the first page that can accommodate the given record. If none of the existing pages can accommodate the given record then add a new page to the file. Assume that the record provided for insertion will always fit into a new blank page.

Search:

Parameter: value of primary key

You are supposed to return the page id and slot id if you find the record in the heap file.

If the record is absent then you should return -1 for both page id and slot id.

Status:

Parameter: none

You should report the number of pages in the heap file, followed by the number of records in each page.

Example input 1:

Line 1 indicates the size of page in bytes for your heap file. Initialize your heap file as an empty file with no pages.

Line 2 asks you to display the status of your heap file. Currently there are zero pages in your heap file.

Line 3 asks you to insert a record of size 20 bytes in the file. Value of primary key is 20. You are expected to add a new page to your heap file. 16 bytes are already consumed to store four book keeping entries. After inserting the record, we have consumed additional 20+4 bytes. Now available empty space in the page is  $100 - (16+24) = 60$ . The largest record now we can accommodate will be of size 56 bytes on this page.

Line 5 asks you to search for record with primary key value 7. There is no such record in the file.

Line 6 asks you to insert a record of size 25 bytes. We will consume 25+4 bytes from the page 0. We will be left with free space of size 31 bytes.

Line 8 asks you to insert record of size 78 bytes. It will require 78+4 bytes. Page 0 cannot accommodate it. We will append a new page and insert the record in page 1. Page 1 now has free space of size only 2 bytes.

Line 11 asks you to quit the program.