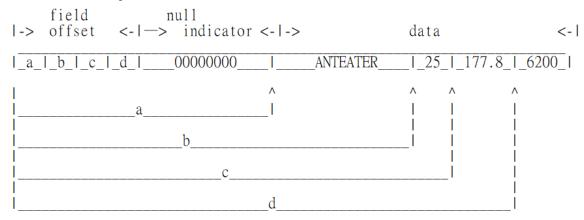
# **Project1 Report**

#### 1. Internal Record Format

- Show your record format design and describe how your design satisfies O(1) field access. If not, just mention that you haven't implemented this feature.
- Describe how you store a VarChar field.



The length of FIELD OFFSET = recordDescriptor.size() \* sizeof(short)

FIELD OFFSET stores the offset of every field. Each of the offset is a short integer which occupies size of (short), that is 2KBs. By reading the value in field offset, we can get information (offset, length) of a field and directly access the field we want with O(1).

e.g. get the 2nd field of data offset = b

length = c - b

we can read the value by starting from offset(b) with length(c-b)

#### 2. Page Format

- Show your page format design

R#n: the n-th record

F: KBs that are used in this page (occupies 2KBs)
M: number of slots in this page (occupies 2KBs)

#n: information of the n-th record, including pointer (2KBs) and length (2KBs)

Note that F starts from 4(KBs) since the initial state (M,F) = (0,4) needs to be written in this page even without any insertion of records. M and F both occupy 2KBs, hence the initial F is 4.

Since records and meta data are inserted from the opposite direction, they will eventually meet each other in the middle of page. However, there may be a space within several KBs that is too small to write data.

### 3. Implementation Detail

R: readPageCounter W: writePageCounter

## - Other implementation details goes here.

When creating a file, the page file manager automatically creates a hidden page that stores readPageCounter, writePageCounter, and appendPageCounter. Every counter is an integer, which occupies 4KBs. Whenever a user writes/reads/appends a page, the corresponding counter + 1. The counter will be stored before closing file.

A: appendPageCounter	
IRIWIAI	-
HIDDEN PAGE	I
	  -
STORE RECORD FROM THIS PAGE	