Name: _____

CS150 Midterm:

Student ID: _____

*You should receive **1 double-sided** answer sheet and an **8-page** exam. You have **100 minutes** (**13:00-14:40**) to complete the exam. Mark your Chinese name and student ID on **both sides of the answer sheet**, **and in the blanks above**. For each question, place **only your final answer** on the answer sheet—do not show work or formulas there. You may use the blank spaces for scratch paper, but **do not tear off any pages**. You will **turn in both question and answer sheets**.*

I. Storage: Disk, Files, Buffers [17 points]

1. [5 points] Write down the letters of true statements *in alphabetical order.* (If none are true, write ∅.)
   A. When querying for a 16 byte record, exactly 16 bytes of data is read from disk.
   B. Writing to an SSD drive is more costly than reading from an SSD drive.
   C. In a heap file, all pages must be filled to capacity except the last page.
   D. If the file size is smaller than the number of buffer frames, a sequential scan of the file using either MRU or LRU (starting with an empty buffer pool) will have the same hit rate.
   E. Assuming integers take 4 bytes and pointers take 4 bytes, a slot directory that is 256 bytes can address 64 records in a page.
   F. In a page containing fixed-length records with no nullable fields, the size of the bitmap never changes.

2. [4 points] Write down the true benefits of using a record header for **variable** length records *in alphabetical order* (or if none are true benefits, write ∅.)
   A. Does not need delimiter character to separate fields in the records.
   B. Always matches or beats space cost when compared to fixed-length record format.
   C. Can access any field without scanning the entire record.
   D. Has compact representation of null value

3. [8 points] Assume we have 4 empty buffer frames and the following access pattern, in which pages are immediately unpinned.

T A M E T E A M M A T E M E A T L I D

Use the replacement policy listed, and list the four pages in the buffer pool at the end, *in alphabetical order.* Hint: you don't need to draw a big chart for every access — look for patterns.

A. MRU.
B. LRU.
C. Clock. (*Assume the clock hand starts on the first buffer and does not move unless a page needs to be replaced.).*

This space left intentionally blank for scratch work.

II. Sort/Hash [16 points]

For this question, consider our table of Products from the previous Joins section, but assume:

- **[P] = 2000 pages**
- **p_C = 100 tuples/page**
- **B = 40 pages** of buffer
- In all parts, we'll use **QuickSort** for our internal sort algorithm.
- **Include the cost of the initial scan *and* cost of writing output in your I/O calculations.**

1. [7 points] First, let's sort our table of Products (P) using the external algorithm we learned in lecture.
   A. How many passes are needed to sort this file?
   B. What is the I/O cost (in pages) of sorting this file?
   C. Suppose we want to decrease the I/O cost of sorting this file, but we want to add the minimum number of buffer pages possible. Among the numbers on the answer sheet (1, 5, 10, 50) circle the *smallest* number of additional buffers to add that decreases the I/O cost.

2. [5 points] Given the resources at the top of Question III, answer the following two questions. Do not bother to simplify arithmetic expressions over constants, like $247*(36^3+\log(4))$.
   A. What is the largest file size (in pages) that we can sort in 3 passes?
   B. What is the smallest file size (in pages) that will require 3 passes to sort?

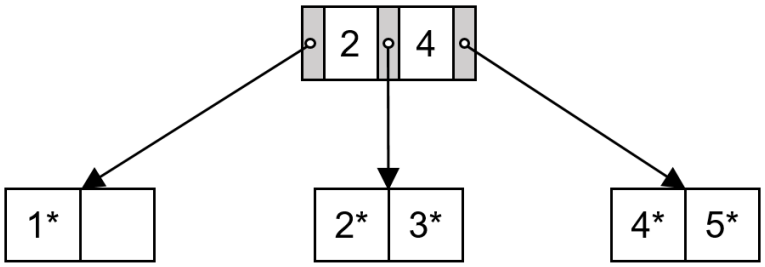
3. [4 points] Suppose I want to eliminate duplicates from our (unsorted) table of Products, using external hashing. Write down the letters of true statements. (If none are true, write $\varnothing$.)
   A. Deduplicating the file using hashing can have a higher IO cost than sorting the file (without deduplicating).
   B. Deduplicating the file using hashing can have a lower IO cost than sorting the file (without deduplicating).
   C. If external hashing recursively partitions on one partition, it will do so on all partitions.

III. Indexes and B+ Trees [18 points]

Note: for B+ tree page splits with an odd number of items, assume that the majority of the items is placed on the right-hand page after the split.

1. [6 points] Alphabetically, write down the letters of statements that apply (or write ∅.)
   A. All internal keys in a B+ tree also appear in its leaf nodes.
   B. The height of a B+ tree increases whenever any node splits.
   C. An ISAM index is similar to a B+ tree, but does not allow for insertion of new values.
   D. The column(s) we select for our index key must have a unique value for every row in the table.
   E. An Alternative 1 index may be either clustered or unclustered.

2. [7 points] The following B+ Tree (height=1) has order 1 (max fanout of 3) and each leaf node can hold up to 2 entries. Answer each of the follow questions **independently of each other**.



   A. What value(s) would be in the root node if we were to insert 0*?
   B. What value(s) would be in the root node if we were to insert 6*?
   C. Starting with the height 1 tree in the picture above, suppose we start inserting keys 6*, 7*, 8*, … and so on. After inserting what key will the height of the tree become 3?

3. [5 points] Assume we are trying to construct a B+ Tree of order 2 (max fanout of 5). Each leaf node can hold up to 4 entries. We insert a total of 16 unique keys via bulk loading, with a fill factor of ¾.
   A. How many leaf nodes will there be?
   B. How many internal (non-leaf) nodes will there be?
   C. How many internal (non-leaf) nodes do we traverse to do an equality search?

IV. SQL [17 points]

Consider the following schema:

```
CREATE TABLE Location (
lname TEXT,
areacode INTEGER PRIMARY KEY)
```

```
CREATE TABLE People(
id INTEGER PRIMARY KEY,
pname TEXT,
areacode INTEGER REFERENCES Location)
```

```
CREATE TABLE Calls(
cid INTEGER PRIMARY KEY,
date DATETIME,
from INTEGER REFERENCES People,
to INTEGER REFERENCES People,
duration INTEGER)
```

You should assume that referential integrity is being enforced, and no NULL values appear.

For parts 1 and 2, fill in the blanks in the SQL queries.

1. [7 points] Names of pairs of people who called each other on 2014-12-25

2. [7 points] Find the location to which the most phone calls have been made, and return its location name as well as the number of calls made to it.

3. [3 points] On the answer sheet, alphabetically write down letters of the statement(s) that find the name of the location of the person who made the longest call. (If none match, write ∅.)

A.
```
WITH (SELECT P.areacode AS areacode, C.duration AS duration
      FROM calls C, People P
      WHERE C.from =P.id) AS DC,
SELECT  L.lname
FROM DC, Location L
WHERE L.areacode = DC.areacode
ORDER BY DC.duration DESC
LIMIT 1;
```

B.
```
SELECT L.lname
FROM Location L, (SELECT P.areacode AS areacode,
                         C.duration AS duration
                  FROM Calls C, People P
                  WHERE C.from =P.id) AS DC
WHERE L.areacode = DC.areacode
ORDER BY DC.duration ASC;
```

C.
```
WITH (SELECT L.lname AS lname, L.areacode AS LAC, P.id AS pid
      FROM Location L FULL OUTER JOIN People P
           ON L.areacode = P.areacode) AS Names,
SELECT Names.lname
FROM Names, Calls C
WHERE C.cid = Names.pid
ORDER BY C.duration DESC
LIMIT 1;
```

V. Joins [17 points]

There are two tables: Candies and Stores.

Candies(candy_id int, candy_name text, manufacturer text)
Stores(store_id int, store_name text, candy_id int)

You want to see which stores carry your favorite candies, but you are unsure of which join algorithm you should utilize to perform the query below.

```
SELECT C.candy_id,
COUNT(*)
FROM Candies C, Stores S
WHERE C.candy_id =
S.candy_id
GROUP BY C.candy_id;
```

We have 100 pages of Candies and 60 pages of Stores. The Candies table has 5 records per page, and the Stores table has 10 records per page. Be sure to choose the inner and outer relations such that you minimize the I/O cost. You have 25 buffer pages at your disposal.

1. [2 points] What is the I/O cost of a block nested loops join between Candies and Stores?

2. [3 points] What is the I/O cost of an index nested loops join between Candies and Stores? (There is an index on Stores.candy_id, and it takes an average of 3 I/O's to find a matching tuple.)

3. [4 points] What is the I/O cost of a sort-merge join between Candies and Stores?

4. [5 points] What is the I/O cost of a grace hash join between Candies and Stores? (Assume we have hash functions that can partition the data evenly.)

5. [3 points] Suppose we add the following clause to the query: "ORDER BY C.candy_id". Which join is best? Mark only one answer.
   A. Block Nested Loop Join
   B. SortMerge
   C. Join
   D. Grace Hash Join
   E. All of the above are equally efficient.

VI. Relational Algebra [15 points]

As shown in the following figures, there are three instances corresponding to three relations: Sailors, Reserves and Boats.

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.0
32	Andy	3	25.0
58	Rusty	10	35.0
74	Horatio	9	35.0

Fig. 1. An instance of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
64	102	9/8/98
74	103	9/8/98

Fig. 2. An instance of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Fig. 3. An instance of Boats

Use relational algebra to describe the following queries.
(Recall that π is project, σ is select, and $\bowtie$ is join.)

1. [2 points] Find the names of sailors who have reserved boat 104.

2. [4 points] Find the names of sailors who have reserved a green boat.

3. [4 points] Find the colors of boats reserved by Dustin.

4. [5 points] Find the names of sailors who have reserved at one boat.