Introduction

By

Krima Doshi, Tanner Greenhagen, Jack Myers, Pavan Chikkathimmegowda,

Shubham Verma, Christian Seto

CSE 510 Database Management Systems Implementation

Arizona State University

6th February 2022

Table of Contents

Abstract 3
Introduction 4
    Terminologies 5
System Requirements 7
    Steps to set up minibase on ubuntu 7
Description of the tests 9
    script command 9
    make test command 9
Conclusion 32
Bibliography 33
Appendix 34

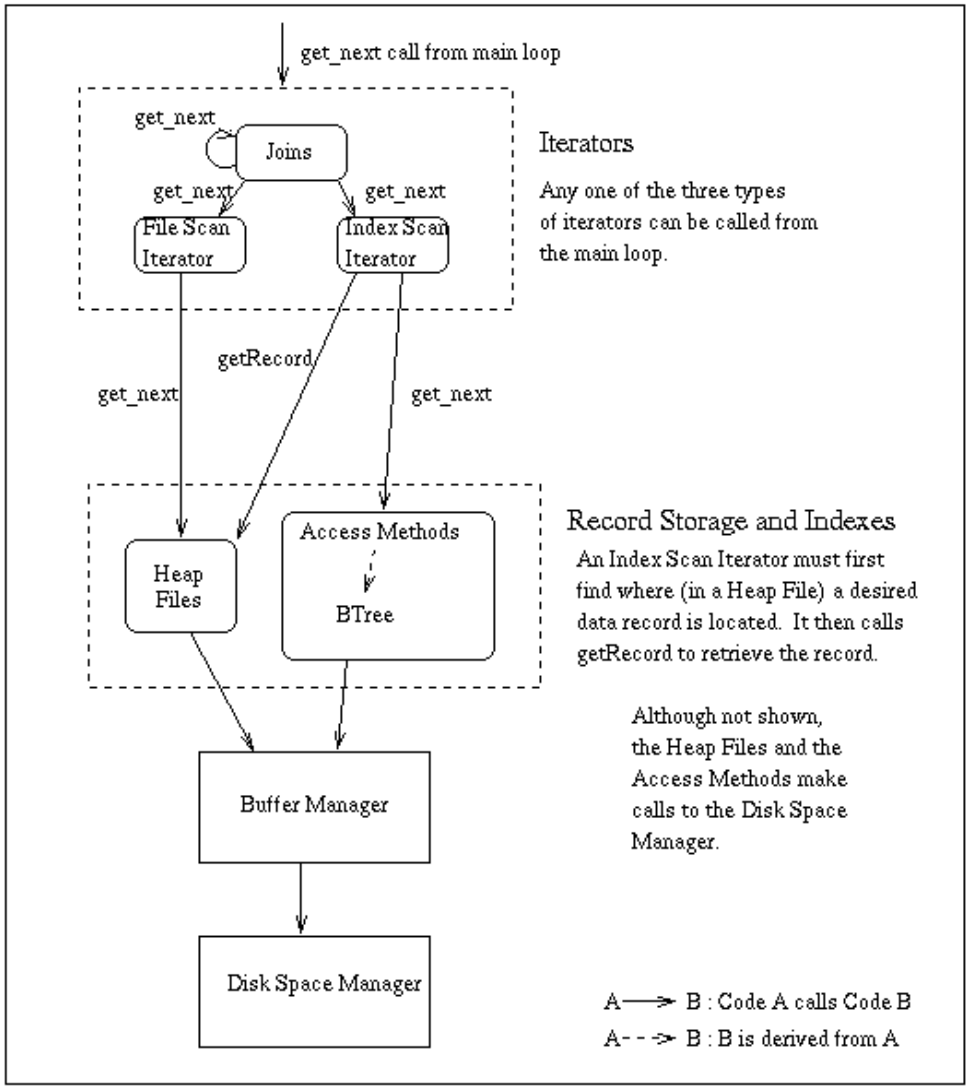
Abstract

Relational Database Systems are one of the most widely used Database Management Systems due to their simplicity in design and good performance. But this brought a different question into consideration, are RDBMS the best solution to the problem? The answer to this question largely varies. When designing for real time applications, there may be other database management options which are a more optimized solution, e.g., Redis database works better on geospatial data which needs to be queried rapidly. The intent of this project is to create one such specialized database management system using the RDBMS database given in minibase.

Keywords: minibase, RDBMS, database management systems implementation, java.

Introduction

Minibase is a small RDMS created using C containing the basic functionalities of a database management system. It's architecture diagram is shown as follows:



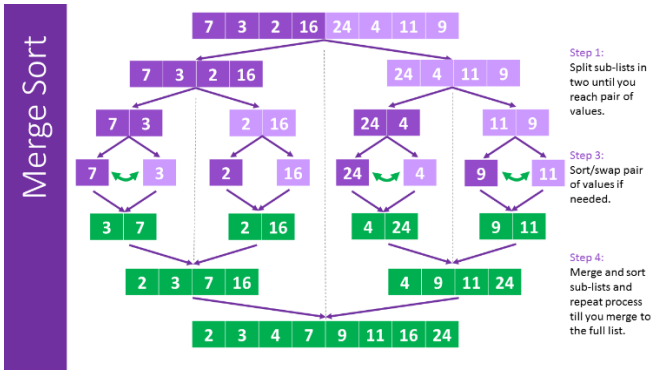
As you can see, it contains a Disk Space Manager in the physical layer. On top of it, it has a buffer manager for loading and caching. Heapfiles and BTrees are created on this buffer manager. IndexScan is then performed on top of this BTree to retrieve files faster using a specified range of records after locating the data record one by one within a specified range. The IndexScan runs in a similar manner on top of heap files. File Scan Iterator iterates over the files one by one for the records (and hence tends to be slower than the IndexScan for records located over multiple files). On top of fetching all these records, comes joins, which deals with joining two tables for querying.

Many other functionalities like concurrency control, lock manager, recovery manager, etc. have been added on later to this minibase. In this project, we use the java version of minibase to implement and further develop an RDBMS system.

Terminologies

Merge Sort

Merge sort is a divide and conquer algorithm. All the records are split recursively into half until one record is obtained and then they are merged one by one recursively. It has O(nlogn) complexity.



In Joins: This algorithm differs from the normal nested loop join. In nested loop join, all the records in the table (having lesser rows) are compared with all the records in the other table and then joined. While in merge sort joins, records are sorted first and merged.

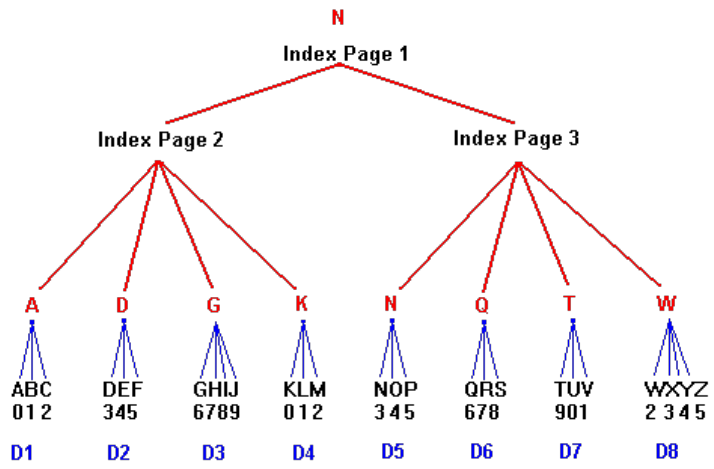
Nested loop joins require less comparisons as compared to merge sort joins. It is good for smaller transactions and data.

Merge sort joins are well suited for large tables and sorted tables.[3]

B-Tree

B+ Trees is a balanced m-ary tree. It contains of a root node, internal nodes, and leaves. In a database, the root node is either a leaf node containing records, or it points to other leaf/internal nodes. Internal nodes are nodes that point to other internal nodes or leaves. Leaves are those nodes which contain the real data, i.e., records of the table.

Search complexity of a B+ tree is O(m*logm(n)). Hence, it helps optimize the performance of the database.



Index Scan

Index Scan is used for index-only accesses in a database. It can be done by specifying the range of records a user want to work on (narrow down the records to scan) and then perform operations using the index of those records directly.

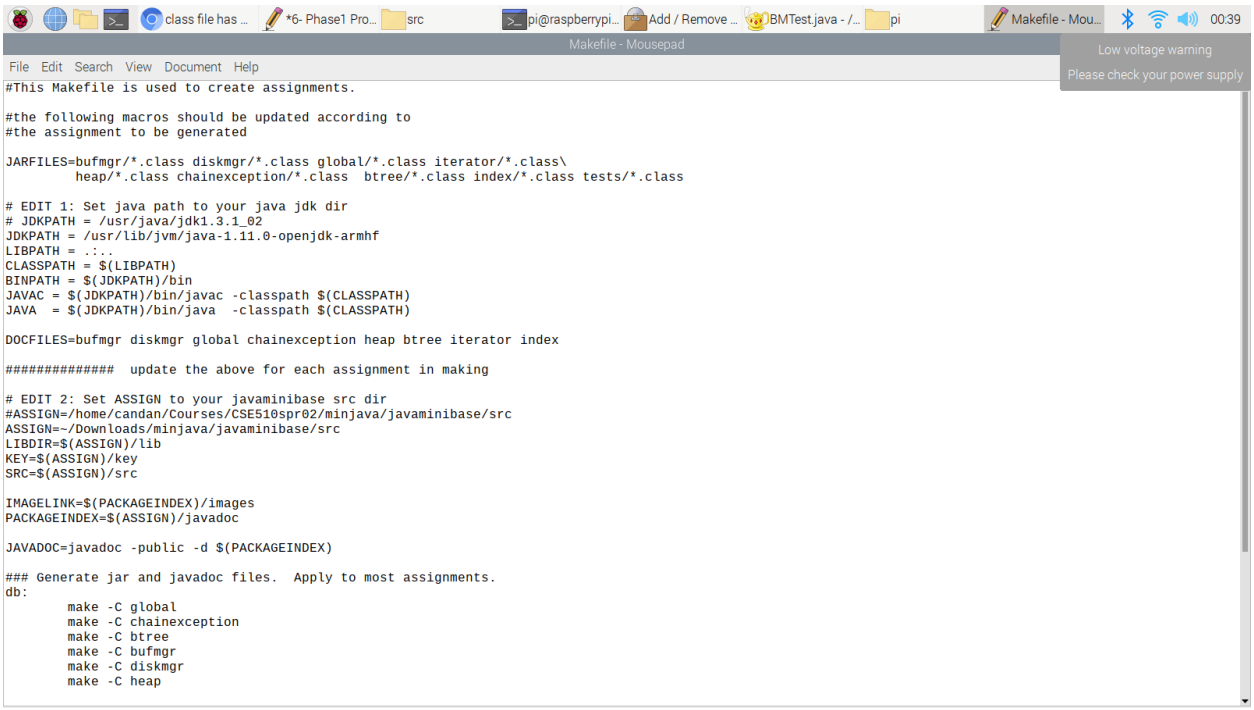
System Requirements

Necessities for running minibase java on your computer include:

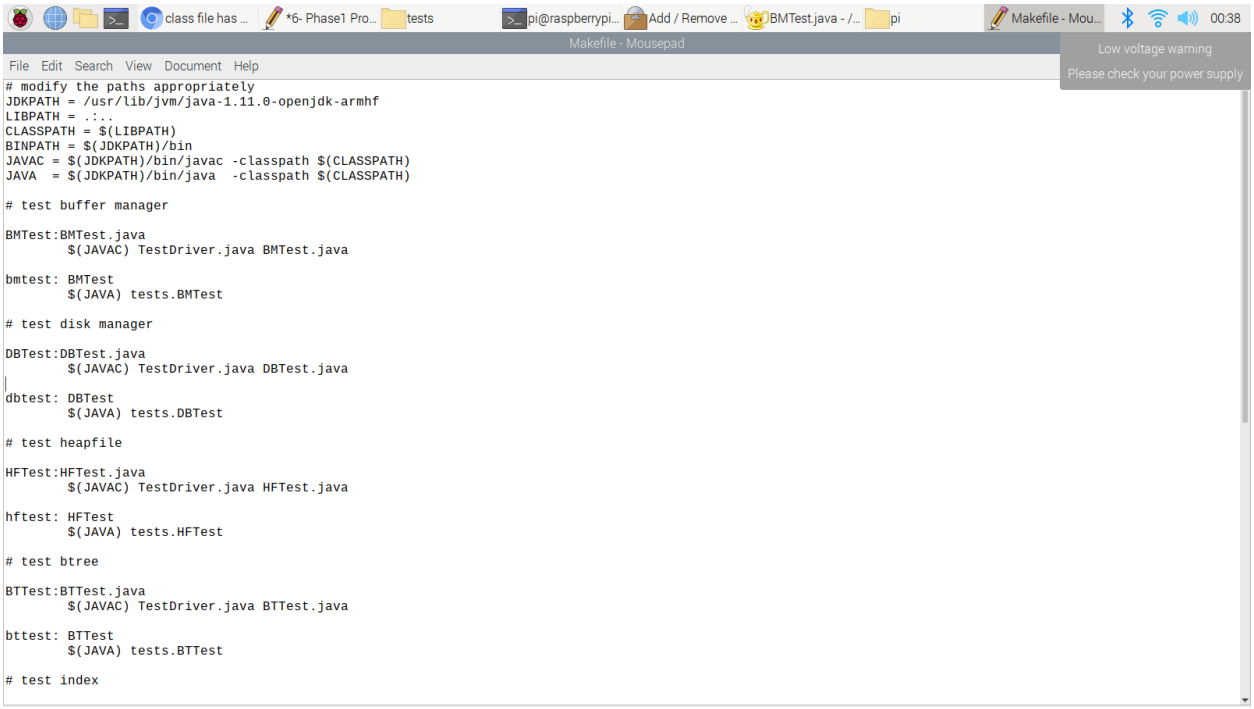
1. An ubuntu OS with java jdk11 installed (JDK version==11 is compatible with the minibase).

Steps to set up minibase on ubuntu

1. Download *minibase.*
2. Extract *minibase.*
3. Set the following variables *in javaminibase/src/Makefile* and *javaminibase/tests/Makefile*:
    a. JDKPATH
    b. ASSIGN




```
#This Makefile is used to create assignments.

#the following macros should be updated according to
#the assignment to be generated

JARFILES=bufmgr/*.class diskmgr/*.class global/*.class iterator/*.class\
         heap/*.class chainexception/*.class  btree/*.class index/*.class tests/*.class

# EDIT 1: Set java path to your java jdk dir
# JDKPATH = /usr/java/jdk1.3.1_02
JDKPATH = /usr/lib/jvm/java-1.11.0-openjdk-armhf
LIBPATH = .:..
CLASSPATH = $(LIBPATH)
BINPATH = $(JDKPATH)/bin
JAVAC = $(JDKPATH)/bin/javac -classpath $(CLASSPATH)
JAVA  = $(JDKPATH)/bin/java  -classpath $(CLASSPATH)

DOCFILES=bufmgr diskmgr global chainexception heap btree iterator index

##############  update the above for each assignment in making

# EDIT 2: Set ASSIGN to your javaminibase src dir
#ASSIGN=/home/candan/Courses/CSE510spr02/minjava/javaminibase/src
ASSIGN=~/Downloads/minjava/javaminibase/src
LIBDIR=$(ASSIGN)/lib
KEY=$(ASSIGN)/key
SRC=$(ASSIGN)/src

IMAGELINK=$(PACKAGEINDEX)/images
PACKAGEINDEX=$(ASSIGN)/javadoc

JAVADOC=javadoc -public -d $(PACKAGEINDEX)

### Generate jar and javadoc files.  Apply to most assignments.
db:
        make -C global
        make -C chainexception
        make -C btree
        make -C bufmgr
        make -C diskmgr
        make -C heap
```




```
# modify the paths appropriately
JDKPATH = /usr/lib/jvm/java-1.11.0-openjdk-armhf
LIBPATH = .:..
CLASSPATH = $(LIBPATH)
BINPATH = $(JDKPATH)/bin
JAVAC = $(JDKPATH)/bin/javac -classpath $(CLASSPATH)
JAVA  = $(JDKPATH)/bin/java  -classpath $(CLASSPATH)

# test buffer manager

BMTest:BMTest.java
        $(JAVAC) TestDriver.java BMTest.java

bmtest: BMTest
        $(JAVA) tests.BMTest

# test disk manager

DBTest:DBTest.java
        $(JAVAC) TestDriver.java DBTest.java

dbtest: DBTest
        $(JAVA) tests.DBTest

# test heapfile

HFTest:HFTest.java
        $(JAVAC) TestDriver.java HFTest.java

hftest: HFTest
        $(JAVA) tests.HFTest

# test btree

BTTest:BTTest.java
        $(JAVAC) TestDriver.java BTTest.java

bttest: BTTest
        $(JAVA) tests.BTTest

# test index
```

4. Remove the following variables from *javaminibase/src/*/Makefile* as they would be inherited from the *Makefile* defined in the *src* folder:
    a. JDKPATH
    b. LIBPATH
5. Run “make db” command from inside *javaminibase/src/Makefile.*

Description of the tests

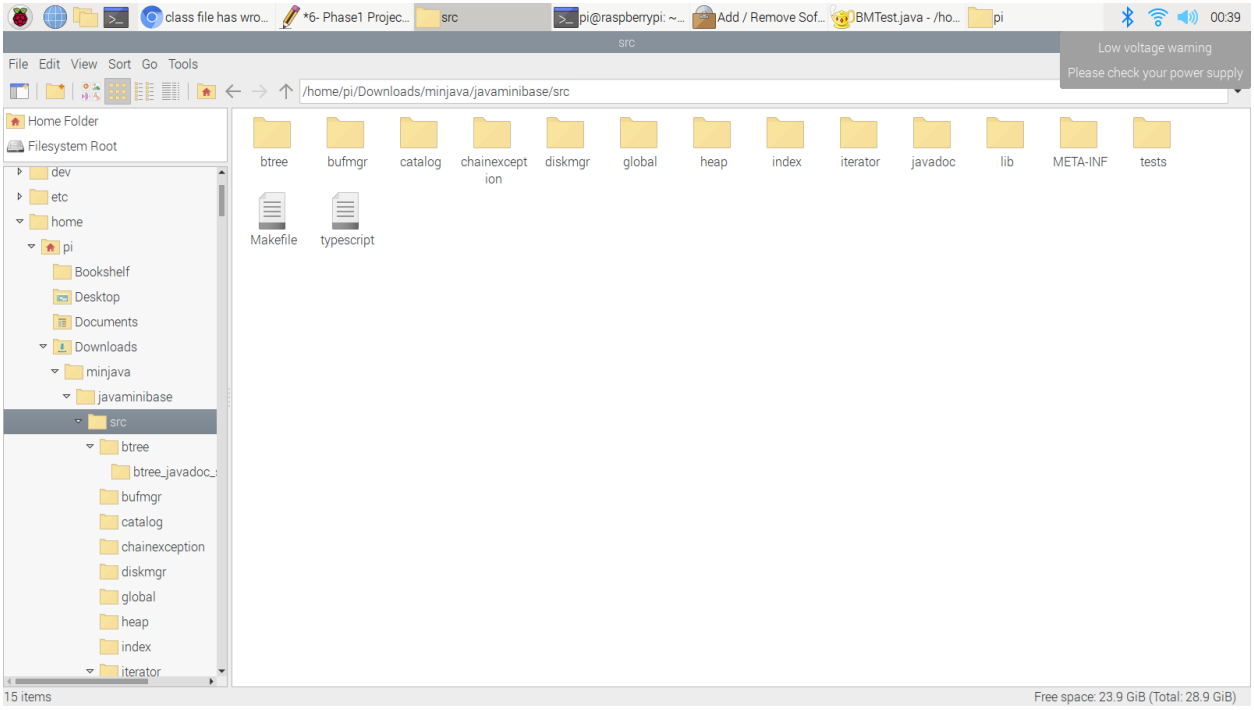
Follow the steps specified in the previous section named “Steps to set up minibase on ubuntu” before proceeding to check the proper execution of the DBMS pipeline. Then, navigate to the *javaminibase/src/* directory and execute the following two commands on your terminal:

1. script
2. make test

After completion of the tests make sure to press CTRL+D so your work is saved.

script command

After running the first command “script”, a file named *typescript* should be automatically created at *javaminibase/src/.*



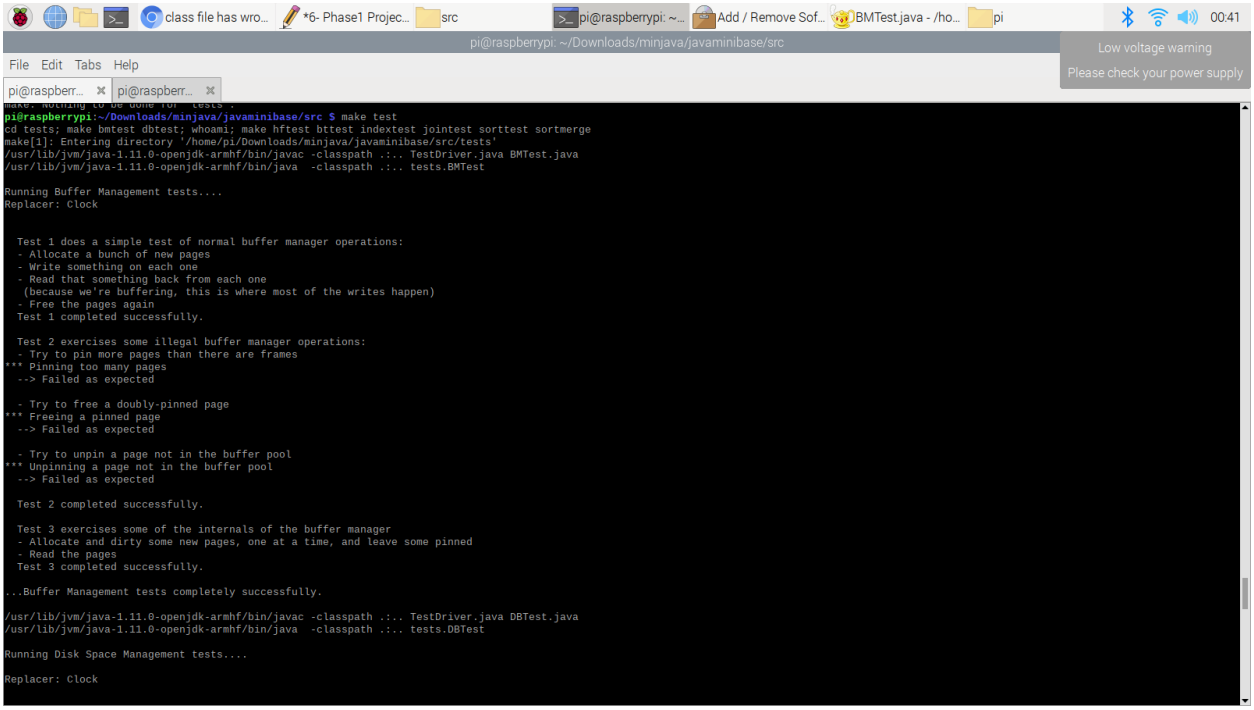
make test command

The make test command will test the SQL statements found in the following files in order as given below:

1. BMTest

The BMTest tests the Buffer Manager. Test 1 runs some simple buffer manager tests. It tries to allocate multiple new pages, unpin the first page from loop, write something on each page (i.e. pin them, write pagenumber+99999 on them and then unpin them), read from each of these pages (pin them, read from

them and unpin them), and free these pages. Test 2 checks if the DBMS can catch illegal budder management ops like pinning more pages than available frames and unpin a page which is not present/made in the buffer pool. Test 3 performs the internal buffer manager testing. It allocates some new pages, write them one at a time and keep them pinned (not unpinning them at the end). Lastly, it checks the read functionality of the page.



2. *DBTest*

This test tests all the execution queries that deal with databases. Test 1 creates a new database by instantiating allocation of a new page, add some file entries to it, write the page and deallocation of the page after completion. Test 2 opens the database created in Test 1, deleting file entries from it, looking up the undeleted file entries, load the content written in Test 1 and finally deallocating the page. Test 3 tests the functioning of error functions by: looking up a deleted entry, deleting an already deleted entry, delete an inexistent file entry, look up an non-existent file entry, adding a duplicate file entry, adding an overflowed entry into a record, run pages that are too long (allocate a very large size to a page), allocate negative page run and deallocate a negative run to pages.

```
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java DBTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.DBTest

Running Disk Space Management tests....

Replacer: Clock


  Test 1 creates a new database and does some tests of normal operations:
  - Add some file entries
  - Allocate a run of pages
  - Write something on some of them
  - Deallocate the rest of them
  Test 1 completed successfully.

  Test 2 opens the database created in test 1 and does some further tests:
  - Delete some of the file entries
  - Look up file entries that should still be there
  - Read stuff back from pages we wrote in test 1
  Test 2 completed successfully.

  Test 3 tests for some error conditions:
  - Look up a deleted file entry
**** Looking up a deleted file entry
  --> Failed as expected

  - Try to delete a deleted entry again
**** Delete a deleted file entry again
  --> Failed as expected

  - Try to delete a nonexistent file entry
**** Deleting a nonexistent file entry
  --> Failed as expected

  - Look up a nonexistent file entry
**** Looking up a nonexistent file entry
  --> Failed as expected

  - Try to add a file entry that's already there
**** Adding a duplicate file entry
  --> Failed as expected

  - Try to add a file entry whose name is too long
**** Adding a file entry with too long a name
  --> Failed as expected

  - Try to allocate a run of pages that's too long
```

```
  --> Failed as expected

  - Try to allocate a run of pages that's too long
**** Allocating a run that's too long
  --> Failed as expected

  - Try to allocate a negative run of pages
**** Allocating a negative run
  --> Failed as expected

  - Try to deallocate a negative run of pages
**** Deallocating a negative run
  --> Failed as expected

  Test 3 completed successfully.

  Test 4 tests some boundary conditions.
    (These tests are very implementation-specific.)
  - Make sure no pages are pinned
  - Allocate all pages remaining after DB overhead is accounted for
  - Attempt to allocate one more page
**** Allocating one additional page
  --> Failed as expected

  - Free some of the allocated pages
  - Allocate some of the just-freed pages
  - Free two continued run of the allocated pages
  - Allocate back number of pages equal to the just freed pages

  - Add enough file entries that the directory must surpass a page
  - Make sure that the directory has taken up an extra page: try to
    allocate more pages than should be available
**** Allocating more pages than are now available
  --> Failed as expected

  - At this point, all pages should be claimed.  Try to allocateone more.
**** Allocating one more page than there is
  --> Failed as expected

  - Free the last two pages: this tests a boundary condition in the space map.
  Test 4 completed successfully.

...Disk Space Management tests completely successfully.

make[1]: Leaving directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
pi
make[1]: Entering directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java HFTest.java
```

3. *HFTest*

These are some tests on the functionalities used on heapfiles. Test 1 tests the creation of a heapfile, inserting records into it, checks if the pages have been pinned to the heapfile for fixed size records. Test 2 tests if fixed-size odd numbered records are properly deleted from the heapfile. Test 3 deals with updating fixed-size records in a heapfile. We use fixed-size records so that there is more predictability

for testing, i.e. for sequential scans, the order differs but for a given fixed-size record, it becomes the order of the number of insertions.

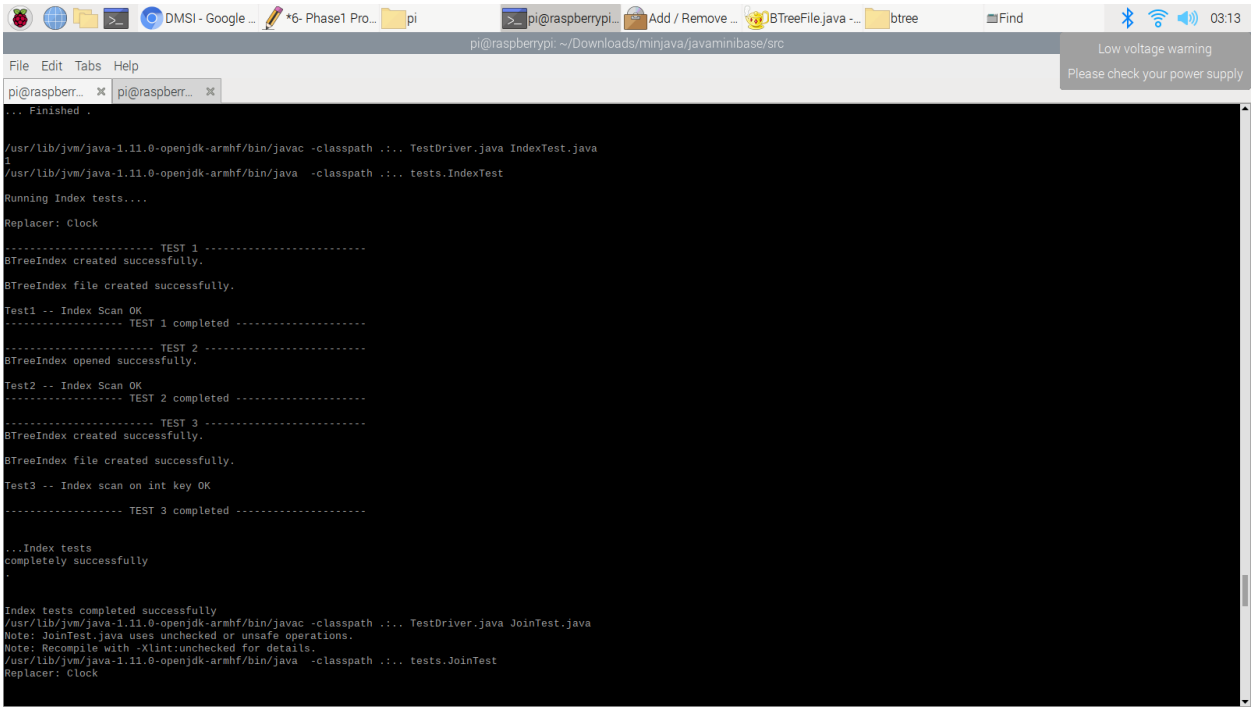
```
make[1]: Entering directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java HFTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.HFTest

Running Heap File tests....

Replacer: Clock


  Test 1: Insert and scan fixed-size records

  - Create a heap file

  - Add 100 records to the file

  - Scan the records just inserted

  Test 1 completed successfully.


  Test 2: Delete fixed-size records

  - Open the same heap file as test 1

  - Delete half the records

  - Scan the remaining records

  Test 2 completed successfully.


  Test 3: Update fixed-size records

  - Open the same heap file as tests 1 and 2

  - Change the records

  - Check that the updates are really there

  Test 3 completed successfully.


  Test 4: Test some error conditions

  - Try to change the size of a record

**** Shortening a record
```

```
  - Try to change the size of a record

**** Shortening a record
  --> Failed as expected

**** Lengthening a record
  --> Failed as expected

  - Try to insert a record that's too long

**** Inserting a too-long record
  --> Failed as expected

  Test 4 completed successfully.


...Heap File tests completely successfully.

/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java BTTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.BTTest
Replacer: Clock


Running  tests....

 ***************** The file name is: AAA0  **********
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

            ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
```

4. *IndexTest*

This test checks the creation of BTreeIndex on the files. Test1 creates a BTreeIndex for a string column file with delete fashion 1 (Full delete[default]). Test 2 creates a String BTreeIndex for a 2-string column

file and creates an IndexScan on the second column. Test 3 creates an Integer BTreeIndex for a 4-column file (2 string, 1 int and 1 real) and creates an IndexScan on the Integer column of the table. IndexScan caches the records in the buffer for quicker access.



5. JoinTest

The JoinTest runs 6 queries to test a simple join, a 3 table join and then adds on conditions to these joined tables. Query 1 simply joins two tables and keeps records having bid value 1. Query 2 joins three tables, keeps records with red color and sorts them by sname. Query 3 joins two tables with a where clause. Query 4 selects distinct values from the joined table. Query 5 selects all ages greater than 40 and ratings less than 7 from the joined tables. Query 6 selects the red color and rating >7 from 3 tables joined together.

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 03:14

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
Replacer: Clock



Any resemblance of persons in this database to people living or dead
is purely coincidental. The contents of this database do not reflect
the views of the University, the Computer  Sciences Department or the
developers...

**********************Query1 strating ********************
Query: Find the names of sailors who have reserved boat number 1.
       and print out the date of reservation.

  SELECT S.sname, R.date
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid AND R.bid = 1


(Tests FileScan, Projection, and Sort-Merge Join)
[Mike Carey, 05/10/95]
[David Dewitt, 05/11/95]
[Jeff Naughton, 05/12/95]

Query1 completed successfully!
*******************Query1 finished!!!*****************



**********************Query2 strating ********************
Query: Find the names of sailors who have reserved a red boat
       and return them in alphabetical order.

  SELECT   S.sname
  FROM     Sailors S, Boats B, Reserves R
  WHERE    S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
  ORDER BY S.sname
Plan used:
 Sort (Pi(sname) (Sigma(B.color='red')  |><|  Pi(sname, bid) (S  |><|  R)))

(Tests File scan, Index scan ,Projection,  index selection,
 sort and simple nested-loop join.)

After Building btree index on sailors.sid.

[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]
[Yannis Ioannidis]
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 03:14

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

**********************Query2 strating ********************
Query: Find the names of sailors who have reserved a red boat
       and return them in alphabetical order.

  SELECT   S.sname
  FROM   Sailors S, Boats B, Reserves R
  WHERE   S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
  ORDER BY S.sname
Plan used:
 Sort (Pi(sname) (Sigma(B.color='red') |><| Pi(sname, bid) (S |><| R)))

(Tests File scan, Index scan ,Projection, index selection,
 sort and simple nested-loop join.)

After Building btree index on sailors.sid.

[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

```
Query2 completed successfully!
*******************Query2 finished!!!*****************



**********************Query3 strating ********************
Query: Find the names of sailors who have reserved a boat.

  SELECT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]
[Mike Carey]
[Mike Carey]
[David Dewitt]
[David Dewitt]
[Jeff Naughton]
[Miron Livny]
[Yannis Ioannidis]
[Raghu Ramakrishnan]
[Raghu Ramakrishnan]
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
*******************Query2 finished!!!*****************



**********************Query3 strating *********************
Query: Find the names of sailors who have reserved a boat.

  SELECT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]
[Mike Carey]
[Mike Carey]
[David Dewitt]
[David Dewitt]
[Jeff Naughton]
[Miron Livny]
[Yannis Ioannidis]
[Raghu Ramakrishnan]
[Raghu Ramakrishnan]

Query3 completed successfully!
*******************Query3 finished!!!*****************



**********************Query4 strating *********************
Query: Find the names of sailors who have reserved a boat
       and print each name once.

  SELECT DISTINCT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]
[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
*******************Query4 finished!!!*****************
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
*******************Query3 finished!!!*****************



**********************Query4 strating *********************
Query: Find the names of sailors who have reserved a boat
       and print each name once.

  SELECT DISTINCT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]
[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
*******************Query4 finished!!!*****************



**********************Query5 strating *********************
Query: Find the names of old sailors or sailors with a rating less
       than 7, who have reserved a boat, (perhaps to increase the
       amount they have to pay to make a reservation).

  SELECT S.sname, S.rating, S.age
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid and (S.age > 40 || S.rating < 7)

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[David Dewitt, 10, 47.2]
[David Dewitt, 10, 47.2]
[Jeff Naughton, 5, 35.0]
[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!
*******************Query5 finished!!!*****************
```

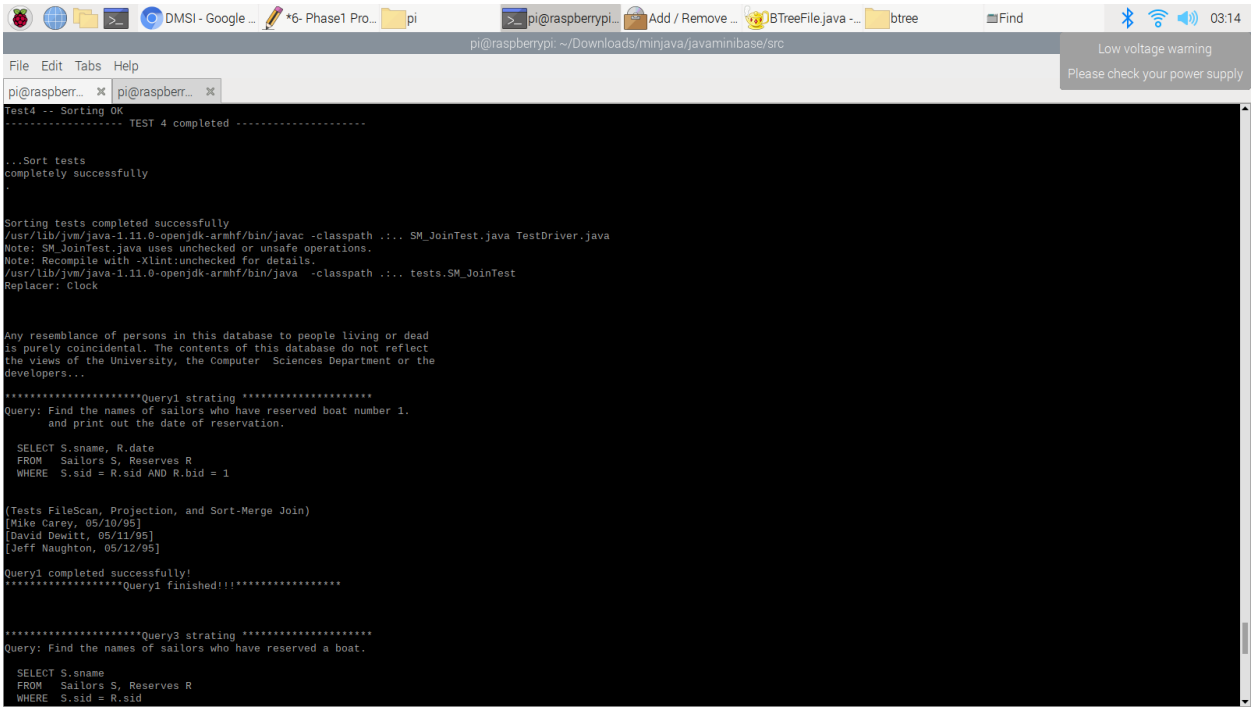
```
**********************Query5 strating *********************
Query: Find the names of old sailors or sailors with a rating less
       than 7, who have reserved a boat, (perhaps to increase the
       amount they have to pay to make a reservation).

  SELECT S.sname, S.rating, S.age
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid and (S.age > 40 || S.rating < 7)

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[David Dewitt, 10, 47.2]
[David Dewitt, 10, 47.2]
[Jeff Naughton, 5, 35.0]
[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!
******************Query5 finished!!!*****************



**********************Query6 strating *********************
Query: Find the names of sailors with a rating greater than 7
  who have reserved a red boat, and print them out in sorted order.

  SELECT   S.sname
  FROM     Sailors S, Boats B, Reserves R
  WHERE    S.sid = R.sid AND S.rating > 7 AND R.bid = B.bid
           AND B.color = 'red'
  ORDER BY S.name

Plan used:
 Sort(Pi(sname) (Sigma(B.color='red')  |><|  Pi(sname, bid) (Sigma(S.rating > 7)  |><|  R)))

(Tests FileScan, Multiple Selection, Projection,sort and nested-loop join.)

After nested loop join S.sid|><|R.sid.
After nested loop join R.bid|><|B.bid AND B.color=red.
After sorting the output tuples.
[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]
[Yannis Ioannidis]
```

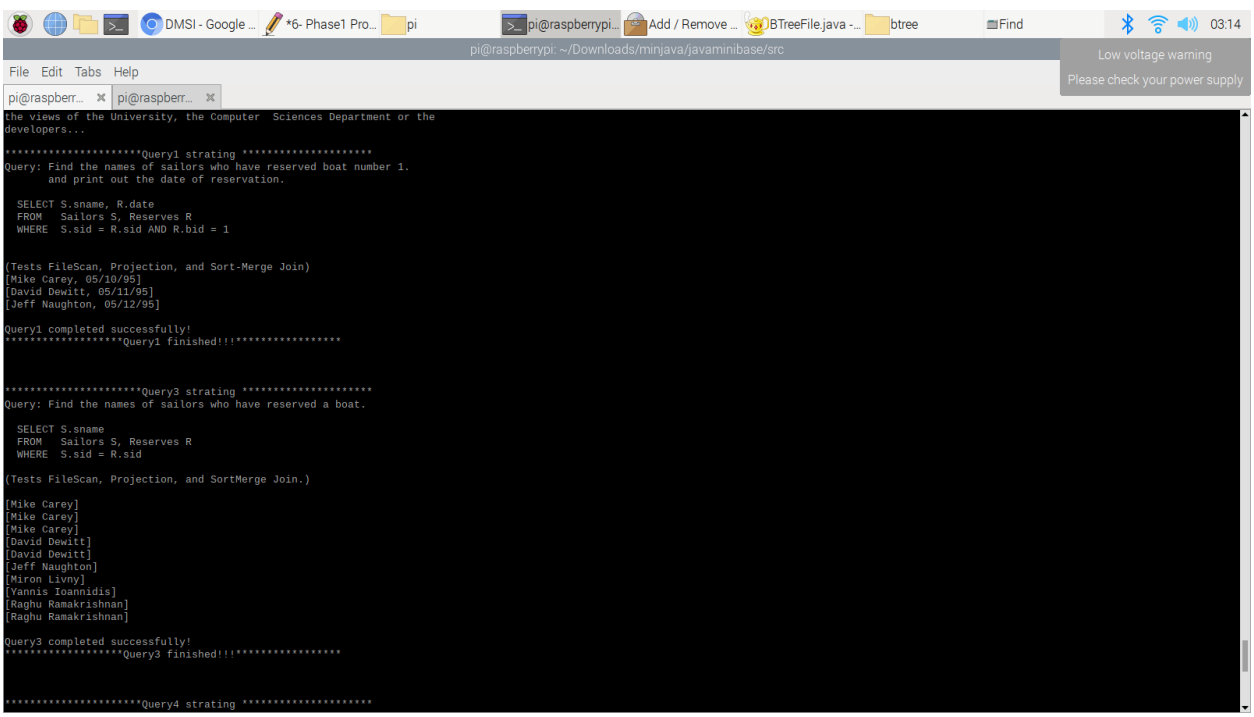
```
Query5 completed successfully!
******************Query5 finished!!!*****************



**********************Query6 strating *********************
Query: Find the names of sailors with a rating greater than 7
  who have reserved a red boat, and print them out in sorted order.

  SELECT   S.sname
  FROM     Sailors S, Boats B, Reserves R
  WHERE    S.sid = R.sid AND S.rating > 7 AND R.bid = B.bid
           AND B.color = 'red'
  ORDER BY S.name

Plan used:
 Sort(Pi(sname) (Sigma(B.color='red')  |><|  Pi(sname, bid) (Sigma(S.rating > 7)  |><|  R)))

(Tests FileScan, Multiple Selection, Projection,sort and nested-loop join.)

After nested loop join S.sid|><|R.sid.
After nested loop join R.bid|><|B.bid AND B.color=red.
After sorting the output tuples.
[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query6 completed successfully!
******************Query6 finished!!!*****************



Finished joins testing
join tests completed successfully
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java SortTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.SortTest

Running Sort tests....

Replacer: Clock

------------------------ TEST 1 -------------------------
Test1 -- Sorting OK
------------------- TEST 1 completed --------------------

------------------------ TEST 2 -------------------------
Test2 -- Sorting OK
```

6. *SM_JoinTest: Sort Merge Join Test*

The SM_JoinTest tests the working of the join using a where clause individually and then goes into a more complex structure using conditions. It executes 6 queries. Query 1 simply uses on condition in the WHERE clause to filter out records and one condition to join the two given tables. Query 2 does not

perform any action. Query 3 joins two tables with a where clause. Query 4 joins two tables with a where clause and selects the DISTINCT values from it. Query 5 selects all ages greater than 40 and ratings less than 7 from the joined tables. Query 6 also does not perform any action.



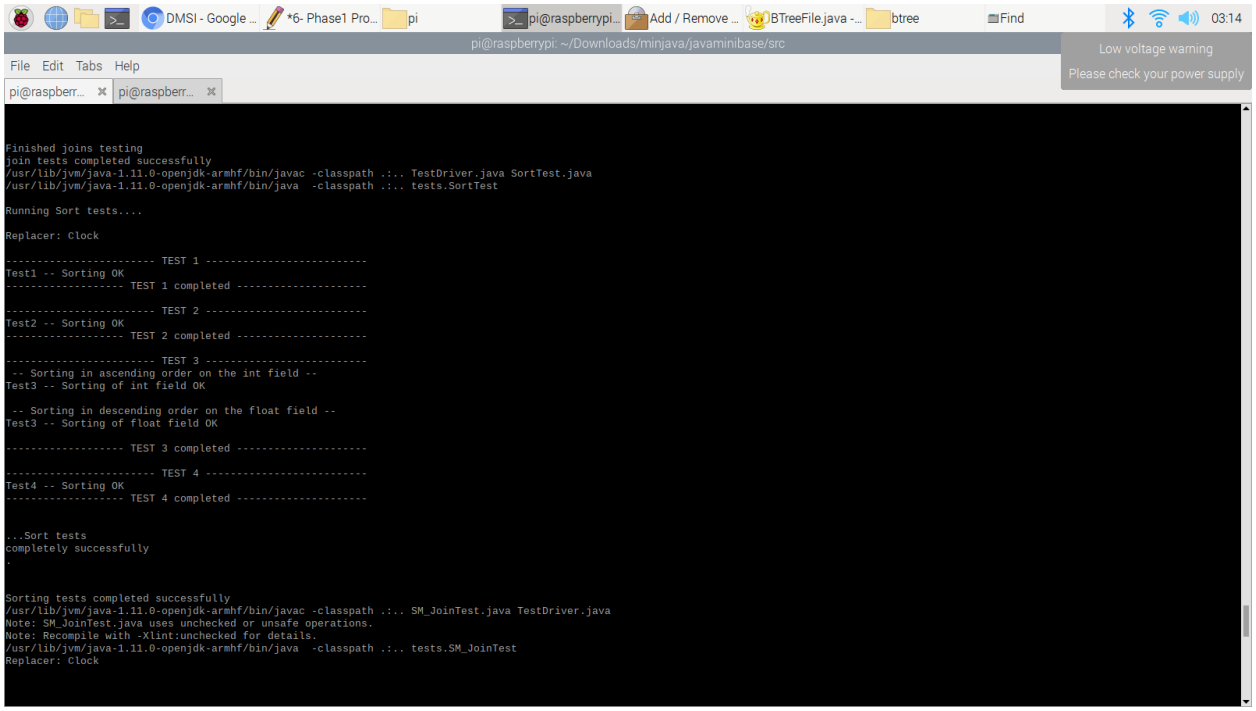


```
**********************Query3 strating *********************
Query: Find the names of sailors who have reserved a boat.

  SELECT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]
[Mike Carey]
[Mike Carey]
[David Dewitt]
[David Dewitt]
[Jeff Naughton]
[Miron Livny]
[Yannis Ioannidis]
[Raghu Ramakrishnan]
[Raghu Ramakrishnan]

Query3 completed successfully!
******************Query3 finished!!!******************



**********************Query4 strating *********************
Query: Find the names of sailors who have reserved a boat
       and print each name once.

  SELECT DISTINCT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]
[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
******************Query4 finished!!!******************
```

```
  SELECT DISTINCT S.sname
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]
[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
******************Query4 finished!!!******************



**********************Query5 strating *********************
Query: Find the names of old sailors or sailors with a rating less
       than 7, who have reserved a boat, (perhaps to increase the
       amount they have to pay to make a reservation).

  SELECT S.sname, S.rating, S.age
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid and (S.age > 40 || S.rating < 7)

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[David Dewitt, 10, 47.2]
[David Dewitt, 10, 47.2]
[Jeff Naughton, 5, 35.0]
[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!
******************Query5 finished!!!******************



Finished joins testing
join tests completed successfully
make[1]: Leaving directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
pi@raspberrypi:~/Downloads/minjava/javaminibase/src $ 1
bash: 1: command not found
pi@raspberrypi:~/Downloads/minjava/javaminibase/src $
```

7. SM_SortTest: Sort Merge Sort Test

This test tests the sorting of an unsorted file. It uses an algorithm known as merge sort. It tests sorting on column 1 of a file having two string columns and specifying Ascending order (Test 1), descending on column 1 in a one string column file (Test 2), sorting in ascending order on the int column 3 of a four column file [two string, one int and on real] (Test 3) and sorting column1 in a two string attribute file both ascending and descending (Test 4).



8. BTTest

This test tests the various functionalities available in a database. It automatically initializes and opens a file named AAA0.

0) Naïve delete

Deletes the first index found for the given key. Naïve delete simply deletes the file. It does not

1) Full delete (Default) (new file)

Deletes all the indices found corresponding to the given key.

2) Print the B+ Tree Structure

Prints the B Tree for the given file.

i. If no records are present on page, it prints “BTree is empty!”
ii. If there is one page, it prints “filenumber pagenumber”
iii. If there are multiple pages, it prints all its “filenumber pagenumber” pairs as leaves and one node for pointing to them on top.

3) Print All Leaf Pages

Prints all the leaf values on the current page.

4) Choose a Page to Print

Prints the content of the page number that is input.

5) **Insert a Record**

Inserts a key onto the current page and reorders the index so that an ascending order is maintained. Allows duplicates.

6) **Delete a Record**

Deletes the first matching record from the current page

i. If key is greater than the largest key on that page or has been deleted already, it does nothing (no error returned).
ii. If key is present, it is deleted.

7) **Test 1 (new file): insert n records in order**

Inserts 0 to n-1 values in order into a new file.

8) **Test 2 (new file): insert n records in reverse order**

Inserts 0 to n-1 values in reverse order into a new file.

9) **Test 3 (new file): insert n records in random order**

Inserts n values in random order into a new file.

10) **Test 4 (new file): insert n records in random order and delete m records randomly**

Inserts n values in random order into a new file and then deleted m records randomly from the same inserted records.

i. If m<=n, it inserts n numbers at random and deletes m
ii. If m>n, it simply deletes all the inserted n records. It does not throw an error.

11) **Delete some records**

Deletes records in between the given keys from the current page. Cases:

i. If key is greater than the largest key on that page, it will delete from the given range until the last. After the last key, it terminates without an error.
ii. If key has been previously deleted is specified in given range, it will delete keys until that key in order, throw an ArrayIndexOutOfBounds for that record and then terminate the deletion process.
iii. If all keys in given range are present on page (i.e., have not been deleted previously), it deletes them successfully.
iv. If no range is specified, it does not delete anything

12) **Initialize a Scan**

This option works whenever there are multiple pages for the given file. It scans the range input by the user from the pages and loads them for quicker access (read and delete, here).

13) **Scan the next Record**

This option scans the next Record from the range specified in 12. If it reaches the end of the range, it prints “AT THE END OF SCAN!”

14) Delete the just-scanned record

This option deletes the record scanned in 13.

i. If no record was scanned previously or reached the end of scan, then it throws a NullPointer error.
ii. Else it deletes that record from the pages

15) Test 5 (new file): insert n records in random order and delete m records randomly

Inserts n string values in random order into a new file and then deleted m records randomly from the same inserted records.

16) Close the file

Closes the current file.

17) Open which file (input an integer for the file name):

Opens the file of the index you input. e.g. for test cases they are 0 for AAA0 (initial file), 1 for AAA1, etc.

i. If file index more than the highest file number you created or has been deleted previously, it throws a pinpage failed - null pointer exception.
ii. If file with index exists, it opens it.

18) Delete which file (input an integer for the file name):

Deletes the file of the index you input. e.g. for test cases they are 0 for AAA0 (initial file), 1 for AAA1, etc.

i. If file index more than the highest file number you created or has been deleted previously, it throws a pinpage failed - null pointer exception
ii. If file with index exists, it deletes it.

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
---------------The B+ Tree Structure---------------
1     3
--------------- End ---------------


------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :0
 ***************** The file name is: AAA1  **********
------------------------- MENU -----------------


[0]   Naive delete (new file)
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
The Tree is Empty!!!
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :1
 ***************** The file name is: AAA2  **********
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:20

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :2



--------------The B+ Tree Structure--------------
1     3
-------------- End --------------


------------------------- MENU -----------------
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:30

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :4
Please input the page number:
3

**************To Print an Leaf Page ********
Current Page ID: 3
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (,  [ 12 12 ]
1 (key, [pageNo, slotNo]):   (,  [ 12 12 ]
************** END ********

------------------------- MENU -----------------
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:20

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :5
Please input the integer key to insert:
12
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:20

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

            ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

            ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages--------------

**************To Print an Leaf Page ********
Current Page ID: 3
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (12,  [ 12 12 ] )
1 (key, [pageNo, slotNo]):   (12,  [ 12 12 ] )
************* END ********
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:31

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
10
 ***************** You open the file: AAA10  **********
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

            ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

            ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :5
Please input the integer key to insert:
1
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
```

```
          ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 19
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (0,  [ 0 0 ] )
1 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
2 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
3 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
4 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

          ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
```

```
------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

          ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
        and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

          ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :6
Please input the integer key to delete:
1
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 19
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (0,  [ 0 0 ] )
1 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
2 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
3 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
************* END ********



------------- All Leaf Pages Have Been Printed --------


------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
[19]  Quit!
Hi, make your choice :1
 ***************** The file name is: AAA2  **********
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :7
Please input the number of keys to insert:
15
 ***************** The file name is: AAA3  **********
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:23

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

pi@raspberr... pi@raspberr...

Low voltage warning
Please check your power supply

```
[13]  Scan the next Record
[14]  Delete the just-scanned record

          ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 7
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (0,  [ 0 0 ] )
1 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
2 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
3 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
4 (key, [pageNo, slotNo]):   (4,  [ 4 4 ] )
5 (key, [pageNo, slotNo]):   (5,  [ 5 5 ] )
6 (key, [pageNo, slotNo]):   (6,  [ 6 6 ] )
7 (key, [pageNo, slotNo]):   (7,  [ 7 7 ] )
8 (key, [pageNo, slotNo]):   (8,  [ 8 8 ] )
9 (key, [pageNo, slotNo]):   (9,  [ 9 9 ] )
10 (key, [pageNo, slotNo]):   (10,  [ 10 10 ] )
11 (key, [pageNo, slotNo]):   (11,  [ 11 11 ] )
12 (key, [pageNo, slotNo]):   (12,  [ 12 12 ] )
13 (key, [pageNo, slotNo]):   (13,  [ 13 13 ] )
14 (key, [pageNo, slotNo]):   (14,  [ 14 14 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU ------------------
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:24

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

pi@raspberr... pi@raspberr...

Low voltage warning
Please check your power supply

```
[19]  Quit!
Hi, make your choice :1
 **************** The file name is: AAA5  **********
-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

          ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

          ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :8
Please input the number of keys to insert:
50
 **************** The file name is: AAA6  **********
-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:24

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
Hi, make your choice :3



---------------The B+ Tree Leaf Pages--------------

**************To Print an Leaf Page ********
Current Page ID: 12
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
1 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
2 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
3 (key, [pageNo, slotNo]):   (4,  [ 4 4 ] )
4 (key, [pageNo, slotNo]):   (5,  [ 5 5 ] )
5 (key, [pageNo, slotNo]):   (6,  [ 6 6 ] )
6 (key, [pageNo, slotNo]):   (7,  [ 7 7 ] )
7 (key, [pageNo, slotNo]):   (8,  [ 8 8 ] )
8 (key, [pageNo, slotNo]):   (9,  [ 9 9 ] )
9 (key, [pageNo, slotNo]):   (10,  [ 10 10 ] )
10 (key, [pageNo, slotNo]):   (11,  [ 11 11 ] )
11 (key, [pageNo, slotNo]):   (12,  [ 12 12 ] )
12 (key, [pageNo, slotNo]):   (13,  [ 13 13 ] )
13 (key, [pageNo, slotNo]):   (14,  [ 14 14 ] )
14 (key, [pageNo, slotNo]):   (15,  [ 15 15 ] )
15 (key, [pageNo, slotNo]):   (16,  [ 16 16 ] )
16 (key, [pageNo, slotNo]):   (17,  [ 17 17 ] )
17 (key, [pageNo, slotNo]):   (18,  [ 18 18 ] )
18 (key, [pageNo, slotNo]):   (19,  [ 19 19 ] )
19 (key, [pageNo, slotNo]):   (20,  [ 20 20 ] )
20 (key, [pageNo, slotNo]):   (21,  [ 21 21 ] )
21 (key, [pageNo, slotNo]):   (22,  [ 22 22 ] )
22 (key, [pageNo, slotNo]):   (23,  [ 23 23 ] )
23 (key, [pageNo, slotNo]):   (24,  [ 24 24 ] )
24 (key, [pageNo, slotNo]):   (25,  [ 25 25 ] )
25 (key, [pageNo, slotNo]):   (26,  [ 26 26 ] )
26 (key, [pageNo, slotNo]):   (27,  [ 27 27 ] )
27 (key, [pageNo, slotNo]):   (28,  [ 28 28 ] )
28 (key, [pageNo, slotNo]):   (29,  [ 29 29 ] )
29 (key, [pageNo, slotNo]):   (30,  [ 30 30 ] )
30 (key, [pageNo, slotNo]):   (31,  [ 31 31 ] )
31 (key, [pageNo, slotNo]):   (32,  [ 32 32 ] )
32 (key, [pageNo, slotNo]):   (33,  [ 33 33 ] )
33 (key, [pageNo, slotNo]):   (34,  [ 34 34 ] )
34 (key, [pageNo, slotNo]):   (35,  [ 35 35 ] )
35 (key, [pageNo, slotNo]):   (36,  [ 36 36 ] )
36 (key, [pageNo, slotNo]):   (37,  [ 37 37 ] )
37 (key, [pageNo, slotNo]):   (38,  [ 38 38 ] )
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:25

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

Low voltage warning
Please check your power supply

pi@raspberr... pi@raspberr...

```
------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :9
Please input the number of keys to insert:
5
 ***************** The file name is: AAA7  **********
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:26

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

pi@raspberr... pi@raspberr...

Low voltage warning
Please check your power supply

```
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 14
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (0,  [ 0 0 ] )
1 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
2 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
3 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
4 (key, [pageNo, slotNo]):   (4,  [ 4 4 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

DMSI - Google ... *6- Phase1 Pro... pi pi@raspberrypi... Add / Remove ... BTreeFile.java -... btree Find 02:32

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

File Edit Tabs Help

pi@raspberr... pi@raspberr...

Low voltage warning
Please check your power supply

```
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

           ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
      and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

           ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :11
Please input the LOWER integer key(>=0):
1
Please input the HIGHER integer key(>=0)
2
-------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)
```

pi@raspberrypi: ~/Downloads/minjava/javaminibase/src

```
[13]  Scan the next Record
[14]  Delete the just-scanned record

          ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
        and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 19
Left Link      : -1
Right Link     : -1
0 (key, [pageNo, slotNo]):   (0,  [ 0 0 ] )
1 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

          ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
```

Conclusion

It is very important to have knowledge of the internal workings of minibase when attempting to implement and convert it into other DBMSs. In this phase, we got familiar with the classes and functionalities present in minibase. We learnt how minibasejava is implemented and how it executes. With the test cases, we got to know the expected behavior of various functionalities of the minibase like: heapfiles, BTreeFiles, BTreeIndex, IndexScan, etc. and their behavior, which will help us further on when implementing our own database.

Bibliography

[1] Minibase Overview. https://research.cs.wisc.edu/coral/minibase/intro/single_user.html. Accessed 2 July 2022.

[2] on, Posted. "Merge Sort Algorithm | 101 Computing." *101 Computing*, 26 Feb. 2019, https://www.101computing.net/merge-sort-algorithm/.

[3] *Difference between Nested Loop join and Sort Merge Join*. https://www.geeksforgeeks.org/difference-between-nested-loop-join-and-sort-merge-join/. Updated: 29 Jul, 2020

[4] *Virtual Machinery Products - BTree Guide*. http://www.virtualmachinery.com/btreeguide.htm. Accessed 2 July 2022.

Appendix

```
Script started on 2022-02-06 15:52:50-07:00 [TERM="xterm-256color" TTY="/dev/pts/0"
COLUMNS="80" LINES="24"]
]0;pi@raspberrypi: ~/Downloads/minjava/javaminibase/src
[01;32mpi@raspberrypi[00m:[01;34m~/Downloads/minjava/javaminibase/src $[00m make test
cd tests; make bmtest dbtest; whoami; make hftest bttest indextest jointest sorttest sortmerge
make[1]: Entering directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java BMTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.BMTest

Running Buffer Management tests....
Replacer: Clock


  Test 1 does a simple test of normal buffer manager operations:
  - Allocate a bunch of new pages
  - Write something on each one
  - Read that something back from each one
   (because we're buffering, this is where most of the writes happen)
  - Free the pages again
  Test 1 completed successfully.

  Test 2 exercises some illegal buffer manager operations:
  - Try to pin more pages than there are frames
*** Pinning too many pages
  --> Failed as expected

  - Try to free a doubly-pinned page
*** Freeing a pinned page
  --> Failed as expected

  - Try to unpin a page not in the buffer pool
*** Unpinning a page not in the buffer pool
  --> Failed as expected

  Test 2 completed successfully.

  Test 3 exercises some of the internals of the buffer manager
  - Allocate and dirty some new pages, one at a time, and leave some pinned
  - Read the pages
  Test 3 completed successfully.

...Buffer Management tests completely successfully.
```

```
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java DBTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.DBTest

Running Disk Space Management tests....

Replacer: Clock


  Test 1 creates a new database and does some tests of normal operations:
  - Add some file entries
  - Allocate a run of pages
  - Write something on some of them
  - Deallocate the rest of them
  Test 1 completed successfully.

  Test 2 opens the database created in test 1 and does some further tests:
  - Delete some of the file entries
  - Look up file entries that should still be there
  - Read stuff back from pages we wrote in test 1
  Test 2 completed successfully.

  Test 3 tests for some error conditions:
  - Look up a deleted file entry
**** Looking up a deleted file entry
  --> Failed as expected

  - Try to delete a deleted entry again
**** Delete a deleted file entry again
  --> Failed as expected

  - Try to delete a nonexistent file entry
**** Deleting a nonexistent file entry
  --> Failed as expected

  - Look up a nonexistent file entry
**** Looking up a nonexistent file entry
  --> Failed as expected

  - Try to add a file entry that's already there
**** Adding a duplicate file entry
  --> Failed as expected

  - Try to add a file entry whose name is too long
**** Adding a file entry with too long a name
```

```
  --> Failed as expected

  - Try to allocate a run of pages that's too long
**** Allocating a run that's too long
  --> Failed as expected

  - Try to allocate a negative run of pages
**** Allocating a negative run
  --> Failed as expected

  - Try to deallocate a negative run of pages
**** Deallocating a negative run
  --> Failed as expected

  Test 3 completed successfully.

  Test 4 tests some boundary conditions.
    (These tests are very implementation-specific.)
  - Make sure no pages are pinned
  - Allocate all pages remaining after DB overhead is accounted for
  - Attempt to allocate one more page
**** Allocating one additional page
  --> Failed as expected

  - Free some of the allocated pages
  - Allocate some of the just-freed pages
  - Free two continued run of the allocated pages
  - Allocate back number of pages equal to the just freed pages

  - Add enough file entries that the directory must surpass a page
  - Make sure that the directory has taken up an extra page: try to
    allocate more pages than should be available
**** Allocating more pages than are now available
   --> Failed as expected

  - At this point, all pages should be claimed.  Try to allocateone more.
**** Allocating one more page than there is
   --> Failed as expected

  - Free the last two pages: this tests a boundary condition in the space map.
  Test 4 completed successfully.

...Disk Space Management tests completely successfully.
```

```
make[1]: Leaving directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
pi
make[1]: Entering directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java HFTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.HFTest

Running Heap File tests....

Replacer: Clock


  Test 1: Insert and scan fixed-size records

  - Create a heap file

  - Add 100 records to the file

  - Scan the records just inserted

  Test 1 completed successfully.


  Test 2: Delete fixed-size records

  - Open the same heap file as test 1

  - Delete half the records

  - Scan the remaining records

  Test 2 completed successfully.


  Test 3: Update fixed-size records

  - Open the same heap file as tests 1 and 2

  - Change the records

  - Check that the updates are really there

  Test 3 completed successfully.
```

```
  Test 4: Test some error conditions

  - Try to change the size of a record

**** Shortening a record
  --> Failed as expected

**** Lengthening a record
  --> Failed as expected

  - Try to insert a record that's too long

**** Inserting a too-long record
  --> Failed as expected

  Test 4 completed successfully.


...Heap File tests completely successfully.

/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java BTTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.BTTest
Replacer: Clock


Running  tests....

 ***************** The file name is: AAA0  **********
-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

          ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
```

```
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :7
Please input the number of keys to insert:
10
 ***************** The file name is: AAA1  **********
------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records
```

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :8
Please input the number of keys to insert:
100
********** The file name is: AAA2 **********
---------- MENU ----------

[0] Naive delete (new file)
[1] Full delete(Default) (new file)

[2] Print the B+ Tree Structure
[3] Print All Leaf Pages
[4] Choose a Page to Print

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record
[6] Delete a Record
[7] Test1 (new file): insert n records in order
[8] Test2 (new file): insert n records in reverse order
[9] Test3 (new file): insert n records in random order
[10] Test4 (new file): insert n records in random order
and delete m records randomly
[11] Delete some records

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :3

----------The B+ Tree Leaf Pages----------

**********To Print an Leaf Page ********
Current Page ID: 6
Left Link : -1
Right Link : 9
0 (key, [pageNo, slotNo]): (1, [1 1])
1 (key, [pageNo, slotNo]): (2, [2 2])
2 (key, [pageNo, slotNo]): (3, [3 3])
3 (key, [pageNo, slotNo]): (4, [4 4])
4 (key, [pageNo, slotNo]): (5, [5 5])
5 (key, [pageNo, slotNo]): (6, [6 6])
6 (key, [pageNo, slotNo]): (7, [7 7])
7 (key, [pageNo, slotNo]): (8, [8 8])
8 (key, [pageNo, slotNo]): (9, [9 9])
9 (key, [pageNo, slotNo]): (10, [10 10])
10 (key, [pageNo, slotNo]): (11, [11 11])
11 (key, [pageNo, slotNo]): (12, [12 12])
12 (key, [pageNo, slotNo]): (13, [13 13])
13 (key, [pageNo, slotNo]): (14, [14 14])
14 (key, [pageNo, slotNo]): (15, [15 15])
15 (key, [pageNo, slotNo]): (16, [16 16])
16 (key, [pageNo, slotNo]): (17, [17 17])
17 (key, [pageNo, slotNo]): (18, [18 18])
18 (key, [pageNo, slotNo]): (19, [19 19])
19 (key, [pageNo, slotNo]): (20, [20 20])
20 (key, [pageNo, slotNo]): (21, [21 21])
21 (key, [pageNo, slotNo]): (22, [22 22])
22 (key, [pageNo, slotNo]): (23, [23 23])
23 (key, [pageNo, slotNo]): (24, [24 24])
24 (key, [pageNo, slotNo]): (25, [25 25])

25 (key, [pageNo, slotNo]): (26, [26 26])
26 (key, [pageNo, slotNo]): (27, [27 27])
27 (key, [pageNo, slotNo]): (28, [28 28])
28 (key, [pageNo, slotNo]): (29, [29 29])
29 (key, [pageNo, slotNo]): (30, [30 30])
30 (key, [pageNo, slotNo]): (31, [31 31])
31 (key, [pageNo, slotNo]): (32, [32 32])
32 (key, [pageNo, slotNo]): (33, [33 33])
33 (key, [pageNo, slotNo]): (34, [34 34])
34 (key, [pageNo, slotNo]): (35, [35 35])
35 (key, [pageNo, slotNo]): (36, [36 36])
36 (key, [pageNo, slotNo]): (37, [37 37])
37 (key, [pageNo, slotNo]): (38, [38 38])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 9
Left Link : 6
Right Link : 7
0 (key, [pageNo, slotNo]): (39, [39 39])
1 (key, [pageNo, slotNo]): (40, [40 40])
2 (key, [pageNo, slotNo]): (41, [41 41])
3 (key, [pageNo, slotNo]): (42, [42 42])
4 (key, [pageNo, slotNo]): (43, [43 43])
5 (key, [pageNo, slotNo]): (44, [44 44])
6 (key, [pageNo, slotNo]): (45, [45 45])
7 (key, [pageNo, slotNo]): (46, [46 46])
8 (key, [pageNo, slotNo]): (47, [47 47])
9 (key, [pageNo, slotNo]): (48, [48 48])
10 (key, [pageNo, slotNo]): (49, [49 49])
11 (key, [pageNo, slotNo]): (50, [50 50])
12 (key, [pageNo, slotNo]): (51, [51 51])
13 (key, [pageNo, slotNo]): (52, [52 52])
14 (key, [pageNo, slotNo]): (53, [53 53])
15 (key, [pageNo, slotNo]): (54, [54 54])
16 (key, [pageNo, slotNo]): (55, [55 55])
17 (key, [pageNo, slotNo]): (56, [56 56])
18 (key, [pageNo, slotNo]): (57, [57 57])
19 (key, [pageNo, slotNo]): (58, [58 58])
20 (key, [pageNo, slotNo]): (59, [59 59])
21 (key, [pageNo, slotNo]): (60, [60 60])
22 (key, [pageNo, slotNo]): (61, [61 61])
23 (key, [pageNo, slotNo]): (62, [62 62])

24 (key, [pageNo, slotNo]): (63, [63 63])
25 (key, [pageNo, slotNo]): (64, [64 64])
26 (key, [pageNo, slotNo]): (65, [65 65])
27 (key, [pageNo, slotNo]): (66, [66 66])
28 (key, [pageNo, slotNo]): (67, [67 67])
29 (key, [pageNo, slotNo]): (68, [68 68])
30 (key, [pageNo, slotNo]): (69, [69 69])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 7
Left Link : 9
Right Link : -1
0 (key, [pageNo, slotNo]): (70, [70 70])
1 (key, [pageNo, slotNo]): (71, [71 71])
2 (key, [pageNo, slotNo]): (72, [72 72])
3 (key, [pageNo, slotNo]): (73, [73 73])
4 (key, [pageNo, slotNo]): (74, [74 74])
5 (key, [pageNo, slotNo]): (75, [75 75])
6 (key, [pageNo, slotNo]): (76, [76 76])
7 (key, [pageNo, slotNo]): (77, [77 77])
8 (key, [pageNo, slotNo]): (78, [78 78])
9 (key, [pageNo, slotNo]): (79, [79 79])
10 (key, [pageNo, slotNo]): (80, [80 80])
11 (key, [pageNo, slotNo]): (81, [81 81])
12 (key, [pageNo, slotNo]): (82, [82 82])
13 (key, [pageNo, slotNo]): (83, [83 83])
14 (key, [pageNo, slotNo]): (84, [84 84])
15 (key, [pageNo, slotNo]): (85, [85 85])
16 (key, [pageNo, slotNo]): (86, [86 86])
17 (key, [pageNo, slotNo]): (87, [87 87])
18 (key, [pageNo, slotNo]): (88, [88 88])
19 (key, [pageNo, slotNo]): (89, [89 89])
20 (key, [pageNo, slotNo]): (90, [90 90])
21 (key, [pageNo, slotNo]): (91, [91 91])
22 (key, [pageNo, slotNo]): (92, [92 92])
23 (key, [pageNo, slotNo]): (93, [93 93])
24 (key, [pageNo, slotNo]): (94, [94 94])
25 (key, [pageNo, slotNo]): (95, [95 95])
26 (key, [pageNo, slotNo]): (96, [96 96])
27 (key, [pageNo, slotNo]): (97, [97 97])
28 (key, [pageNo, slotNo]): (98, [98 98])
29 (key, [pageNo, slotNo]): (99, [99 99])

```
30 (key, [pageNo, slotNo]):   (100,  [ 100 100 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :2
```

```
----------------The B+ Tree Structure----------------
1    8
2         6
2         9
2         7
---------------- End ----------------


-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):
```

```
[19]  Quit!
Hi, make your choice :12
Please input the LOWER integer key (null if -3):
20
Please input the HIGHER integer key (null if -2):
20
------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :13
SCAN RESULT: 20 [ 20 20 ]
```

```
------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



---------------The B+ Tree Leaf Pages---------------

**************To Print an Leaf Page ********
Current Page ID: 6
Left Link      : -1
```

Right Link : 9
0 (key, [pageNo, slotNo]): (1, [1 1])
1 (key, [pageNo, slotNo]): (2, [2 2])
2 (key, [pageNo, slotNo]): (3, [3 3])
3 (key, [pageNo, slotNo]): (4, [4 4])
4 (key, [pageNo, slotNo]): (5, [5 5])
5 (key, [pageNo, slotNo]): (6, [6 6])
6 (key, [pageNo, slotNo]): (7, [7 7])
7 (key, [pageNo, slotNo]): (8, [8 8])
8 (key, [pageNo, slotNo]): (9, [9 9])
9 (key, [pageNo, slotNo]): (10, [10 10])
10 (key, [pageNo, slotNo]): (11, [11 11])
11 (key, [pageNo, slotNo]): (12, [12 12])
12 (key, [pageNo, slotNo]): (13, [13 13])
13 (key, [pageNo, slotNo]): (14, [14 14])
14 (key, [pageNo, slotNo]): (15, [15 15])
15 (key, [pageNo, slotNo]): (16, [16 16])
16 (key, [pageNo, slotNo]): (17, [17 17])
17 (key, [pageNo, slotNo]): (18, [18 18])
18 (key, [pageNo, slotNo]): (19, [19 19])
19 (key, [pageNo, slotNo]): (20, [20 20])
20 (key, [pageNo, slotNo]): (21, [21 21])
21 (key, [pageNo, slotNo]): (22, [22 22])
22 (key, [pageNo, slotNo]): (23, [23 23])
23 (key, [pageNo, slotNo]): (24, [24 24])
24 (key, [pageNo, slotNo]): (25, [25 25])
25 (key, [pageNo, slotNo]): (26, [26 26])
26 (key, [pageNo, slotNo]): (27, [27 27])
27 (key, [pageNo, slotNo]): (28, [28 28])
28 (key, [pageNo, slotNo]): (29, [29 29])
29 (key, [pageNo, slotNo]): (30, [30 30])
30 (key, [pageNo, slotNo]): (31, [31 31])
31 (key, [pageNo, slotNo]): (32, [32 32])
32 (key, [pageNo, slotNo]): (33, [33 33])
33 (key, [pageNo, slotNo]): (34, [34 34])
34 (key, [pageNo, slotNo]): (35, [35 35])
35 (key, [pageNo, slotNo]): (36, [36 36])
36 (key, [pageNo, slotNo]): (37, [37 37])
37 (key, [pageNo, slotNo]): (38, [38 38])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 9

Left Link : 6
Right Link : 7
0 (key, [pageNo, slotNo]): (39, [39 39])
1 (key, [pageNo, slotNo]): (40, [40 40])
2 (key, [pageNo, slotNo]): (41, [41 41])
3 (key, [pageNo, slotNo]): (42, [42 42])
4 (key, [pageNo, slotNo]): (43, [43 43])
5 (key, [pageNo, slotNo]): (44, [44 44])
6 (key, [pageNo, slotNo]): (45, [45 45])
7 (key, [pageNo, slotNo]): (46, [46 46])
8 (key, [pageNo, slotNo]): (47, [47 47])
9 (key, [pageNo, slotNo]): (48, [48 48])
10 (key, [pageNo, slotNo]): (49, [49 49])
11 (key, [pageNo, slotNo]): (50, [50 50])
12 (key, [pageNo, slotNo]): (51, [51 51])
13 (key, [pageNo, slotNo]): (52, [52 52])
14 (key, [pageNo, slotNo]): (53, [53 53])
15 (key, [pageNo, slotNo]): (54, [54 54])
16 (key, [pageNo, slotNo]): (55, [55 55])
17 (key, [pageNo, slotNo]): (56, [56 56])
18 (key, [pageNo, slotNo]): (57, [57 57])
19 (key, [pageNo, slotNo]): (58, [58 58])
20 (key, [pageNo, slotNo]): (59, [59 59])
21 (key, [pageNo, slotNo]): (60, [60 60])
22 (key, [pageNo, slotNo]): (61, [61 61])
23 (key, [pageNo, slotNo]): (62, [62 62])
24 (key, [pageNo, slotNo]): (63, [63 63])
25 (key, [pageNo, slotNo]): (64, [64 64])
26 (key, [pageNo, slotNo]): (65, [65 65])
27 (key, [pageNo, slotNo]): (66, [66 66])
28 (key, [pageNo, slotNo]): (67, [67 67])
29 (key, [pageNo, slotNo]): (68, [68 68])
30 (key, [pageNo, slotNo]): (69, [69 69])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 7
Left Link : 9
Right Link : -1
0 (key, [pageNo, slotNo]): (70, [70 70])
1 (key, [pageNo, slotNo]): (71, [71 71])
2 (key, [pageNo, slotNo]): (72, [72 72])
3 (key, [pageNo, slotNo]): (73, [73 73])

```
4 (key, [pageNo, slotNo]):   (74,  [ 74 74 ] )
5 (key, [pageNo, slotNo]):   (75,  [ 75 75 ] )
6 (key, [pageNo, slotNo]):   (76,  [ 76 76 ] )
7 (key, [pageNo, slotNo]):   (77,  [ 77 77 ] )
8 (key, [pageNo, slotNo]):   (78,  [ 78 78 ] )
9 (key, [pageNo, slotNo]):   (79,  [ 79 79 ] )
10 (key, [pageNo, slotNo]):   (80,  [ 80 80 ] )
11 (key, [pageNo, slotNo]):   (81,  [ 81 81 ] )
12 (key, [pageNo, slotNo]):   (82,  [ 82 82 ] )
13 (key, [pageNo, slotNo]):   (83,  [ 83 83 ] )
14 (key, [pageNo, slotNo]):   (84,  [ 84 84 ] )
15 (key, [pageNo, slotNo]):   (85,  [ 85 85 ] )
16 (key, [pageNo, slotNo]):   (86,  [ 86 86 ] )
17 (key, [pageNo, slotNo]):   (87,  [ 87 87 ] )
18 (key, [pageNo, slotNo]):   (88,  [ 88 88 ] )
19 (key, [pageNo, slotNo]):   (89,  [ 89 89 ] )
20 (key, [pageNo, slotNo]):   (90,  [ 90 90 ] )
21 (key, [pageNo, slotNo]):   (91,  [ 91 91 ] )
22 (key, [pageNo, slotNo]):   (92,  [ 92 92 ] )
23 (key, [pageNo, slotNo]):   (93,  [ 93 93 ] )
24 (key, [pageNo, slotNo]):   (94,  [ 94 94 ] )
25 (key, [pageNo, slotNo]):   (95,  [ 95 95 ] )
26 (key, [pageNo, slotNo]):   (96,  [ 96 96 ] )
27 (key, [pageNo, slotNo]):   (97,  [ 97 97 ] )
28 (key, [pageNo, slotNo]):   (98,  [ 98 98 ] )
29 (key, [pageNo, slotNo]):   (99,  [ 99 99 ] )
30 (key, [pageNo, slotNo]):   (100,  [ 100 100 ] )
************** END ********


------------- All Leaf Pages Have Been Printed --------


------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print
```

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record
[6] Delete a Record
[7] Test1 (new file): insert n records in order
[8] Test2 (new file): insert n records in reverse order
[9] Test3 (new file): insert n records in random order
[10] Test4 (new file): insert n records in random order
and delete m records randomly
[11] Delete some records

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :14
---------- MENU ----------

[0] Naive delete (new file)
[1] Full delete(Default) (new file)

[2] Print the B+ Tree Structure
[3] Print All Leaf Pages
[4] Choose a Page to Print

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record
[6] Delete a Record
[7] Test1 (new file): insert n records in order
[8] Test2 (new file): insert n records in reverse order
[9] Test3 (new file): insert n records in random order
[10] Test4 (new file): insert n records in random order

and delete m records randomly
[11] Delete some records

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :3

----------The B+ Tree Leaf Pages----------

**********To Print an Leaf Page ********
Current Page ID: 6
Left Link : -1
Right Link : 9
0 (key, [pageNo, slotNo]): (1, [1 1])
1 (key, [pageNo, slotNo]): (2, [2 2])
2 (key, [pageNo, slotNo]): (3, [3 3])
3 (key, [pageNo, slotNo]): (4, [4 4])
4 (key, [pageNo, slotNo]): (5, [5 5])
5 (key, [pageNo, slotNo]): (6, [6 6])
6 (key, [pageNo, slotNo]): (7, [7 7])
7 (key, [pageNo, slotNo]): (8, [8 8])
8 (key, [pageNo, slotNo]): (9, [9 9])
9 (key, [pageNo, slotNo]): (10, [10 10])
10 (key, [pageNo, slotNo]): (11, [11 11])
11 (key, [pageNo, slotNo]): (12, [12 12])
12 (key, [pageNo, slotNo]): (13, [13 13])
13 (key, [pageNo, slotNo]): (14, [14 14])
14 (key, [pageNo, slotNo]): (15, [15 15])
15 (key, [pageNo, slotNo]): (16, [16 16])
16 (key, [pageNo, slotNo]): (17, [17 17])

17 (key, [pageNo, slotNo]): (18, [18 18])
18 (key, [pageNo, slotNo]): (19, [19 19])
19 (key, [pageNo, slotNo]): (21, [21 21])
20 (key, [pageNo, slotNo]): (22, [22 22])
21 (key, [pageNo, slotNo]): (23, [23 23])
22 (key, [pageNo, slotNo]): (24, [24 24])
23 (key, [pageNo, slotNo]): (25, [25 25])
24 (key, [pageNo, slotNo]): (26, [26 26])
25 (key, [pageNo, slotNo]): (27, [27 27])
26 (key, [pageNo, slotNo]): (28, [28 28])
27 (key, [pageNo, slotNo]): (29, [29 29])
28 (key, [pageNo, slotNo]): (30, [30 30])
29 (key, [pageNo, slotNo]): (31, [31 31])
30 (key, [pageNo, slotNo]): (32, [32 32])
31 (key, [pageNo, slotNo]): (33, [33 33])
32 (key, [pageNo, slotNo]): (34, [34 34])
33 (key, [pageNo, slotNo]): (35, [35 35])
34 (key, [pageNo, slotNo]): (36, [36 36])
35 (key, [pageNo, slotNo]): (37, [37 37])
36 (key, [pageNo, slotNo]): (38, [38 38])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 9
Left Link : 6
Right Link : 7
0 (key, [pageNo, slotNo]): (39, [39 39])
1 (key, [pageNo, slotNo]): (40, [40 40])
2 (key, [pageNo, slotNo]): (41, [41 41])
3 (key, [pageNo, slotNo]): (42, [42 42])
4 (key, [pageNo, slotNo]): (43, [43 43])
5 (key, [pageNo, slotNo]): (44, [44 44])
6 (key, [pageNo, slotNo]): (45, [45 45])
7 (key, [pageNo, slotNo]): (46, [46 46])
8 (key, [pageNo, slotNo]): (47, [47 47])
9 (key, [pageNo, slotNo]): (48, [48 48])
10 (key, [pageNo, slotNo]): (49, [49 49])
11 (key, [pageNo, slotNo]): (50, [50 50])
12 (key, [pageNo, slotNo]): (51, [51 51])
13 (key, [pageNo, slotNo]): (52, [52 52])
14 (key, [pageNo, slotNo]): (53, [53 53])
15 (key, [pageNo, slotNo]): (54, [54 54])
16 (key, [pageNo, slotNo]): (55, [55 55])

17 (key, [pageNo, slotNo]): (56, [56 56])
18 (key, [pageNo, slotNo]): (57, [57 57])
19 (key, [pageNo, slotNo]): (58, [58 58])
20 (key, [pageNo, slotNo]): (59, [59 59])
21 (key, [pageNo, slotNo]): (60, [60 60])
22 (key, [pageNo, slotNo]): (61, [61 61])
23 (key, [pageNo, slotNo]): (62, [62 62])
24 (key, [pageNo, slotNo]): (63, [63 63])
25 (key, [pageNo, slotNo]): (64, [64 64])
26 (key, [pageNo, slotNo]): (65, [65 65])
27 (key, [pageNo, slotNo]): (66, [66 66])
28 (key, [pageNo, slotNo]): (67, [67 67])
29 (key, [pageNo, slotNo]): (68, [68 68])
30 (key, [pageNo, slotNo]): (69, [69 69])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 7
Left Link : 9
Right Link : -1
0 (key, [pageNo, slotNo]): (70, [70 70])
1 (key, [pageNo, slotNo]): (71, [71 71])
2 (key, [pageNo, slotNo]): (72, [72 72])
3 (key, [pageNo, slotNo]): (73, [73 73])
4 (key, [pageNo, slotNo]): (74, [74 74])
5 (key, [pageNo, slotNo]): (75, [75 75])
6 (key, [pageNo, slotNo]): (76, [76 76])
7 (key, [pageNo, slotNo]): (77, [77 77])
8 (key, [pageNo, slotNo]): (78, [78 78])
9 (key, [pageNo, slotNo]): (79, [79 79])
10 (key, [pageNo, slotNo]): (80, [80 80])
11 (key, [pageNo, slotNo]): (81, [81 81])
12 (key, [pageNo, slotNo]): (82, [82 82])
13 (key, [pageNo, slotNo]): (83, [83 83])
14 (key, [pageNo, slotNo]): (84, [84 84])
15 (key, [pageNo, slotNo]): (85, [85 85])
16 (key, [pageNo, slotNo]): (86, [86 86])
17 (key, [pageNo, slotNo]): (87, [87 87])
18 (key, [pageNo, slotNo]): (88, [88 88])
19 (key, [pageNo, slotNo]): (89, [89 89])
20 (key, [pageNo, slotNo]): (90, [90 90])
21 (key, [pageNo, slotNo]): (91, [91 91])
22 (key, [pageNo, slotNo]): (92, [92 92])

```
23 (key, [pageNo, slotNo]):   (93,  [ 93 93 ] )
24 (key, [pageNo, slotNo]):   (94,  [ 94 94 ] )
25 (key, [pageNo, slotNo]):   (95,  [ 95 95 ] )
26 (key, [pageNo, slotNo]):   (96,  [ 96 96 ] )
27 (key, [pageNo, slotNo]):   (97,  [ 97 97 ] )
28 (key, [pageNo, slotNo]):   (98,  [ 98 98 ] )
29 (key, [pageNo, slotNo]):   (99,  [ 99 99 ] )
30 (key, [pageNo, slotNo]):   (100,  [ 100 100 ] )
************** END ********



------------- All Leaf Pages Have Been Printed --------


------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.
```

```
[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :12
Please input the LOWER integer key (null if -3):
30
Please input the HIGHER integer key (null if -2):
50
------------------------- MENU -----------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):
```

```
[19]  Quit!
Hi, make your choice :13
SCAN RESULT: 30 [ 30 30 ]
------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :14
------------------------- MENU ------------------


[0]   Naive delete (new file)
```

```
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :3



----------------The B+ Tree Leaf Pages----------------

**************To Print an Leaf Page ********
Current Page ID: 6
Left Link      : -1
Right Link     : 9
0 (key, [pageNo, slotNo]):   (1,  [ 1 1 ] )
1 (key, [pageNo, slotNo]):   (2,  [ 2 2 ] )
2 (key, [pageNo, slotNo]):   (3,  [ 3 3 ] )
```

3 (key, [pageNo, slotNo]): (4, [4 4])
4 (key, [pageNo, slotNo]): (5, [5 5])
5 (key, [pageNo, slotNo]): (6, [6 6])
6 (key, [pageNo, slotNo]): (7, [7 7])
7 (key, [pageNo, slotNo]): (8, [8 8])
8 (key, [pageNo, slotNo]): (9, [9 9])
9 (key, [pageNo, slotNo]): (10, [10 10])
10 (key, [pageNo, slotNo]): (11, [11 11])
11 (key, [pageNo, slotNo]): (12, [12 12])
12 (key, [pageNo, slotNo]): (13, [13 13])
13 (key, [pageNo, slotNo]): (14, [14 14])
14 (key, [pageNo, slotNo]): (15, [15 15])
15 (key, [pageNo, slotNo]): (16, [16 16])
16 (key, [pageNo, slotNo]): (17, [17 17])
17 (key, [pageNo, slotNo]): (18, [18 18])
18 (key, [pageNo, slotNo]): (19, [19 19])
19 (key, [pageNo, slotNo]): (21, [21 21])
20 (key, [pageNo, slotNo]): (22, [22 22])
21 (key, [pageNo, slotNo]): (23, [23 23])
22 (key, [pageNo, slotNo]): (24, [24 24])
23 (key, [pageNo, slotNo]): (25, [25 25])
24 (key, [pageNo, slotNo]): (26, [26 26])
25 (key, [pageNo, slotNo]): (27, [27 27])
26 (key, [pageNo, slotNo]): (28, [28 28])
27 (key, [pageNo, slotNo]): (29, [29 29])
28 (key, [pageNo, slotNo]): (31, [31 31])
29 (key, [pageNo, slotNo]): (32, [32 32])
30 (key, [pageNo, slotNo]): (33, [33 33])
31 (key, [pageNo, slotNo]): (34, [34 34])
32 (key, [pageNo, slotNo]): (35, [35 35])
33 (key, [pageNo, slotNo]): (36, [36 36])
34 (key, [pageNo, slotNo]): (37, [37 37])
35 (key, [pageNo, slotNo]): (38, [38 38])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 9
Left Link : 6
Right Link : 7
0 (key, [pageNo, slotNo]): (39, [39 39])
1 (key, [pageNo, slotNo]): (40, [40 40])
2 (key, [pageNo, slotNo]): (41, [41 41])
3 (key, [pageNo, slotNo]): (42, [42 42])

```
4 (key, [pageNo, slotNo]):   (43,  [ 43 43 ] )
5 (key, [pageNo, slotNo]):   (44,  [ 44 44 ] )
6 (key, [pageNo, slotNo]):   (45,  [ 45 45 ] )
7 (key, [pageNo, slotNo]):   (46,  [ 46 46 ] )
8 (key, [pageNo, slotNo]):   (47,  [ 47 47 ] )
9 (key, [pageNo, slotNo]):   (48,  [ 48 48 ] )
10 (key, [pageNo, slotNo]):   (49,  [ 49 49 ] )
11 (key, [pageNo, slotNo]):   (50,  [ 50 50 ] )
12 (key, [pageNo, slotNo]):   (51,  [ 51 51 ] )
13 (key, [pageNo, slotNo]):   (52,  [ 52 52 ] )
14 (key, [pageNo, slotNo]):   (53,  [ 53 53 ] )
15 (key, [pageNo, slotNo]):   (54,  [ 54 54 ] )
16 (key, [pageNo, slotNo]):   (55,  [ 55 55 ] )
17 (key, [pageNo, slotNo]):   (56,  [ 56 56 ] )
18 (key, [pageNo, slotNo]):   (57,  [ 57 57 ] )
19 (key, [pageNo, slotNo]):   (58,  [ 58 58 ] )
20 (key, [pageNo, slotNo]):   (59,  [ 59 59 ] )
21 (key, [pageNo, slotNo]):   (60,  [ 60 60 ] )
22 (key, [pageNo, slotNo]):   (61,  [ 61 61 ] )
23 (key, [pageNo, slotNo]):   (62,  [ 62 62 ] )
24 (key, [pageNo, slotNo]):   (63,  [ 63 63 ] )
25 (key, [pageNo, slotNo]):   (64,  [ 64 64 ] )
26 (key, [pageNo, slotNo]):   (65,  [ 65 65 ] )
27 (key, [pageNo, slotNo]):   (66,  [ 66 66 ] )
28 (key, [pageNo, slotNo]):   (67,  [ 67 67 ] )
29 (key, [pageNo, slotNo]):   (68,  [ 68 68 ] )
30 (key, [pageNo, slotNo]):   (69,  [ 69 69 ] )
************** END ********


**************To Print an Leaf Page ********
Current Page ID: 7
Left Link      : 9
Right Link     : -1
0 (key, [pageNo, slotNo]):   (70,  [ 70 70 ] )
1 (key, [pageNo, slotNo]):   (71,  [ 71 71 ] )
2 (key, [pageNo, slotNo]):   (72,  [ 72 72 ] )
3 (key, [pageNo, slotNo]):   (73,  [ 73 73 ] )
4 (key, [pageNo, slotNo]):   (74,  [ 74 74 ] )
5 (key, [pageNo, slotNo]):   (75,  [ 75 75 ] )
6 (key, [pageNo, slotNo]):   (76,  [ 76 76 ] )
7 (key, [pageNo, slotNo]):   (77,  [ 77 77 ] )
8 (key, [pageNo, slotNo]):   (78,  [ 78 78 ] )
9 (key, [pageNo, slotNo]):   (79,  [ 79 79 ] )
```

```
10 (key, [pageNo, slotNo]):   (80,  [ 80 80 ] )
11 (key, [pageNo, slotNo]):   (81,  [ 81 81 ] )
12 (key, [pageNo, slotNo]):   (82,  [ 82 82 ] )
13 (key, [pageNo, slotNo]):   (83,  [ 83 83 ] )
14 (key, [pageNo, slotNo]):   (84,  [ 84 84 ] )
15 (key, [pageNo, slotNo]):   (85,  [ 85 85 ] )
16 (key, [pageNo, slotNo]):   (86,  [ 86 86 ] )
17 (key, [pageNo, slotNo]):   (87,  [ 87 87 ] )
18 (key, [pageNo, slotNo]):   (88,  [ 88 88 ] )
19 (key, [pageNo, slotNo]):   (89,  [ 89 89 ] )
20 (key, [pageNo, slotNo]):   (90,  [ 90 90 ] )
21 (key, [pageNo, slotNo]):   (91,  [ 91 91 ] )
22 (key, [pageNo, slotNo]):   (92,  [ 92 92 ] )
23 (key, [pageNo, slotNo]):   (93,  [ 93 93 ] )
24 (key, [pageNo, slotNo]):   (94,  [ 94 94 ] )
25 (key, [pageNo, slotNo]):   (95,  [ 95 95 ] )
26 (key, [pageNo, slotNo]):   (96,  [ 96 96 ] )
27 (key, [pageNo, slotNo]):   (97,  [ 97 97 ] )
28 (key, [pageNo, slotNo]):   (98,  [ 98 98 ] )
29 (key, [pageNo, slotNo]):   (99,  [ 99 99 ] )
30 (key, [pageNo, slotNo]):   (100,  [ 100 100 ] )
************** END ********


------------- All Leaf Pages Have Been Printed --------


-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

        ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
```

```
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records

[12]  Initialize a Scan
[13]  Scan the next Record
[14]  Delete the just-scanned record

         ---String Key (for choice [15]) ---

[15]  Test5 (new file): insert n records in random order
       and delete m records randomly.

[16]  Close the file
[17]  Open which file (input an integer for the file name):
[18]  Destroy which file (input an integer for the file name):

[19]  Quit!
Hi, make your choice :12
Please input the LOWER integer key (null if -3):
40
Please input the HIGHER integer key (null if -2):
50
-------------------------- MENU ------------------


[0]   Naive delete (new file)
[1]   Full delete(Default) (new file)

[2]   Print the B+ Tree Structure
[3]   Print All Leaf Pages
[4]   Choose a Page to Print

         ---Integer Key (for choices [6]-[14]) ---

[5]   Insert a Record
[6]   Delete a Record
[7]   Test1 (new file): insert n records in order
[8]   Test2 (new file): insert n records in reverse order
[9]   Test3 (new file): insert n records in random order
[10]  Test4 (new file): insert n records in random order
     and delete m records randomly
[11]  Delete some records
```

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :14
---------- MENU ----------

[0] Naive delete (new file)
[1] Full delete(Default) (new file)

[2] Print the B+ Tree Structure
[3] Print All Leaf Pages
[4] Choose a Page to Print

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record
[6] Delete a Record
[7] Test1 (new file): insert n records in order
[8] Test2 (new file): insert n records in reverse order
[9] Test3 (new file): insert n records in random order
[10] Test4 (new file): insert n records in random order
and delete m records randomly
[11] Delete some records

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order

and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :3

----------The B+ Tree Leaf Pages----------

**********To Print an Leaf Page ********
Current Page ID: 6
Left Link : -1
Right Link : 9
0 (key, [pageNo, slotNo]): (1, [1 1])
1 (key, [pageNo, slotNo]): (2, [2 2])
2 (key, [pageNo, slotNo]): (3, [3 3])
3 (key, [pageNo, slotNo]): (4, [4 4])
4 (key, [pageNo, slotNo]): (5, [5 5])
5 (key, [pageNo, slotNo]): (6, [6 6])
6 (key, [pageNo, slotNo]): (7, [7 7])
7 (key, [pageNo, slotNo]): (8, [8 8])
8 (key, [pageNo, slotNo]): (9, [9 9])
9 (key, [pageNo, slotNo]): (10, [10 10])
10 (key, [pageNo, slotNo]): (11, [11 11])
11 (key, [pageNo, slotNo]): (12, [12 12])
12 (key, [pageNo, slotNo]): (13, [13 13])
13 (key, [pageNo, slotNo]): (14, [14 14])
14 (key, [pageNo, slotNo]): (15, [15 15])
15 (key, [pageNo, slotNo]): (16, [16 16])
16 (key, [pageNo, slotNo]): (17, [17 17])
17 (key, [pageNo, slotNo]): (18, [18 18])
18 (key, [pageNo, slotNo]): (19, [19 19])
19 (key, [pageNo, slotNo]): (21, [21 21])
20 (key, [pageNo, slotNo]): (22, [22 22])
21 (key, [pageNo, slotNo]): (23, [23 23])
22 (key, [pageNo, slotNo]): (24, [24 24])
23 (key, [pageNo, slotNo]): (25, [25 25])
24 (key, [pageNo, slotNo]): (26, [26 26])
25 (key, [pageNo, slotNo]): (27, [27 27])
26 (key, [pageNo, slotNo]): (28, [28 28])

27 (key, [pageNo, slotNo]): (29, [29 29])
28 (key, [pageNo, slotNo]): (31, [31 31])
29 (key, [pageNo, slotNo]): (32, [32 32])
30 (key, [pageNo, slotNo]): (33, [33 33])
31 (key, [pageNo, slotNo]): (34, [34 34])
32 (key, [pageNo, slotNo]): (35, [35 35])
33 (key, [pageNo, slotNo]): (36, [36 36])
34 (key, [pageNo, slotNo]): (37, [37 37])
35 (key, [pageNo, slotNo]): (38, [38 38])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 9
Left Link : 6
Right Link : 7
0 (key, [pageNo, slotNo]): (39, [39 39])
1 (key, [pageNo, slotNo]): (40, [40 40])
2 (key, [pageNo, slotNo]): (41, [41 41])
3 (key, [pageNo, slotNo]): (42, [42 42])
4 (key, [pageNo, slotNo]): (43, [43 43])
5 (key, [pageNo, slotNo]): (44, [44 44])
6 (key, [pageNo, slotNo]): (45, [45 45])
7 (key, [pageNo, slotNo]): (46, [46 46])
8 (key, [pageNo, slotNo]): (47, [47 47])
9 (key, [pageNo, slotNo]): (48, [48 48])
10 (key, [pageNo, slotNo]): (49, [49 49])
11 (key, [pageNo, slotNo]): (50, [50 50])
12 (key, [pageNo, slotNo]): (51, [51 51])
13 (key, [pageNo, slotNo]): (52, [52 52])
14 (key, [pageNo, slotNo]): (53, [53 53])
15 (key, [pageNo, slotNo]): (54, [54 54])
16 (key, [pageNo, slotNo]): (55, [55 55])
17 (key, [pageNo, slotNo]): (56, [56 56])
18 (key, [pageNo, slotNo]): (57, [57 57])
19 (key, [pageNo, slotNo]): (58, [58 58])
20 (key, [pageNo, slotNo]): (59, [59 59])
21 (key, [pageNo, slotNo]): (60, [60 60])
22 (key, [pageNo, slotNo]): (61, [61 61])
23 (key, [pageNo, slotNo]): (62, [62 62])
24 (key, [pageNo, slotNo]): (63, [63 63])
25 (key, [pageNo, slotNo]): (64, [64 64])
26 (key, [pageNo, slotNo]): (65, [65 65])
27 (key, [pageNo, slotNo]): (66, [66 66])

28 (key, [pageNo, slotNo]): (67, [67 67])
29 (key, [pageNo, slotNo]): (68, [68 68])
30 (key, [pageNo, slotNo]): (69, [69 69])
********** END ********

**********To Print an Leaf Page ********
Current Page ID: 7
Left Link : 9
Right Link : -1
0 (key, [pageNo, slotNo]): (70, [70 70])
1 (key, [pageNo, slotNo]): (71, [71 71])
2 (key, [pageNo, slotNo]): (72, [72 72])
3 (key, [pageNo, slotNo]): (73, [73 73])
4 (key, [pageNo, slotNo]): (74, [74 74])
5 (key, [pageNo, slotNo]): (75, [75 75])
6 (key, [pageNo, slotNo]): (76, [76 76])
7 (key, [pageNo, slotNo]): (77, [77 77])
8 (key, [pageNo, slotNo]): (78, [78 78])
9 (key, [pageNo, slotNo]): (79, [79 79])
10 (key, [pageNo, slotNo]): (80, [80 80])
11 (key, [pageNo, slotNo]): (81, [81 81])
12 (key, [pageNo, slotNo]): (82, [82 82])
13 (key, [pageNo, slotNo]): (83, [83 83])
14 (key, [pageNo, slotNo]): (84, [84 84])
15 (key, [pageNo, slotNo]): (85, [85 85])
16 (key, [pageNo, slotNo]): (86, [86 86])
17 (key, [pageNo, slotNo]): (87, [87 87])
18 (key, [pageNo, slotNo]): (88, [88 88])
19 (key, [pageNo, slotNo]): (89, [89 89])
20 (key, [pageNo, slotNo]): (90, [90 90])
21 (key, [pageNo, slotNo]): (91, [91 91])
22 (key, [pageNo, slotNo]): (92, [92 92])
23 (key, [pageNo, slotNo]): (93, [93 93])
24 (key, [pageNo, slotNo]): (94, [94 94])
25 (key, [pageNo, slotNo]): (95, [95 95])
26 (key, [pageNo, slotNo]): (96, [96 96])
27 (key, [pageNo, slotNo]): (97, [97 97])
28 (key, [pageNo, slotNo]): (98, [98 98])
29 (key, [pageNo, slotNo]): (99, [99 99])
30 (key, [pageNo, slotNo]): (100, [100 100])
********** END ********

---------- All Leaf Pages Have Been Printed --------

---------- MENU ----------

[0] Naive delete (new file)
[1] Full delete(Default) (new file)

[2] Print the B+ Tree Structure
[3] Print All Leaf Pages
[4] Choose a Page to Print

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record
[6] Delete a Record
[7] Test1 (new file): insert n records in order
[8] Test2 (new file): insert n records in reverse order
[9] Test3 (new file): insert n records in random order
[10] Test4 (new file): insert n records in random order
and delete m records randomly
[11] Delete some records

[12] Initialize a Scan
[13] Scan the next Record
[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order
and delete m records randomly.

[16] Close the file
[17] Open which file (input an integer for the file name):
[18] Destroy which file (input an integer for the file name):

[19] Quit!
Hi, make your choice :19

... Finished .

```
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java IndexTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.IndexTest

Running Index tests....

Replacer: Clock

------------------------ TEST 1 -------------------------
BTreeIndex created successfully.

BTreeIndex file created successfully.

Test1 -- Index Scan OK
------------------- TEST 1 completed ---------------------

------------------------ TEST 2 -------------------------
BTreeIndex opened successfully.

Test2 -- Index Scan OK
------------------- TEST 2 completed ---------------------

------------------------ TEST 3 -------------------------
BTreeIndex created successfully.

BTreeIndex file created successfully.

Test3 -- Index scan on int key OK

------------------- TEST 3 completed ---------------------


...Index tests
completely successfully
.


Index tests completed successfully
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java JoinTest.java
Note: JoinTest.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.JoinTest
Replacer: Clock
```

Any resemblance of persons in this database to people living or dead
is purely coincidental. The contents of this database do not reflect
the views of the University, the Computer Sciences Department or the
developers...

**********Query1 strating **********
Query: Find the names of sailors who have reserved boat number 1.
    and print out the date of reservation.

```
SELECT S.sname, R.date
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid AND R.bid = 1
```

(Tests FileScan, Projection, and Sort-Merge Join)
[Mike Carey, 05/10/95]
[David Dewitt, 05/11/95]
[Jeff Naughton, 05/12/95]

Query1 completed successfully!
**********Query1 finished!!!**********

**********Query2 strating **********
Query: Find the names of sailors who have reserved a red boat
    and return them in alphabetical order.

```
SELECT   S.sname
FROM     Sailors S, Boats B, Reserves R
WHERE    S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
ORDER BY S.sname
```

Plan used:
Sort (Pi(sname) (Sigma(B.color='red') |><| Pi(sname, bid) (S |><| R)))

(Tests File scan, Index scan ,Projection, index selection,
sort and simple nested-loop join.)

After Building btree index on sailors.sid.

[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]

[Yannis Ioannidis]

Query2 completed successfully!
**********Query2 finished!!!**********

**********Query3 strating **********
Query: Find the names of sailors who have reserved a boat.

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
```

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]
[Mike Carey]
[Mike Carey]
[David Dewitt]
[David Dewitt]
[Jeff Naughton]
[Miron Livny]
[Yannis Ioannidis]
[Raghu Ramakrishnan]
[Raghu Ramakrishnan]

Query3 completed successfully!
**********Query3 finished!!!**********

**********Query4 strating **********
Query: Find the names of sailors who have reserved a boat
and print each name once.

```
SELECT DISTINCT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
```

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]

[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
**********Query4 finished!!!**********

**********Query5 strating **********
Query: Find the names of old sailors or sailors with a rating less
than 7, who have reserved a boat, (perhaps to increase the
amount they have to pay to make a reservation).

```
SELECT S.sname, S.rating, S.age
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid and (S.age > 40 || S.rating < 7)
```

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[David Dewitt, 10, 47.2]
[David Dewitt, 10, 47.2]
[Jeff Naughton, 5, 35.0]
[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!
**********Query5 finished!!!**********

**********Query6 strating **********
Query: Find the names of sailors with a rating greater than 7
who have reserved a red boat, and print them out in sorted order.

```
SELECT   S.sname
FROM     Sailors S, Boats B, Reserves R
WHERE    S.sid = R.sid AND S.rating > 7 AND R.bid = B.bid
         AND B.color = 'red'
ORDER BY S.name
```

```
Plan used:
 Sort(Pi(sname) (Sigma(B.color='red')  |><|  Pi(sname, bid) (Sigma(S.rating > 7)  |><|  R)))

(Tests FileScan, Multiple Selection, Projection,sort and nested-loop join.)

After nested loop join S.sid|><|R.sid.
After nested loop join R.bid|><|B.bid AND B.color=red.
After sorting the output tuples.
[David Dewitt]
[Mike Carey]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query6 completed successfully!
*******************Query6 finished!!!*****************



Finished joins testing
join tests completed successfully
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. TestDriver.java SortTest.java
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.SortTest

Running Sort tests....

Replacer: Clock

------------------------- TEST 1 --------------------------
Test1 -- Sorting OK
------------------- TEST 1 completed ---------------------

------------------------- TEST 2 --------------------------
Test2 -- Sorting OK
------------------- TEST 2 completed ---------------------

------------------------- TEST 3 --------------------------
 -- Sorting in ascending order on the int field --
Test3 -- Sorting of int field OK

 -- Sorting in descending order on the float field --
Test3 -- Sorting of float field OK

------------------- TEST 3 completed ---------------------
```

```
------------------------ TEST 4 -------------------------
Test4 -- Sorting OK
------------------- TEST 4 completed ---------------------


...Sort tests
completely successfully
.


Sorting tests completed successfully
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/javac -classpath .:.. SM_JoinTest.java TestDriver.java
Note: SM_JoinTest.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
/usr/lib/jvm/java-1.11.0-openjdk-armhf/bin/java  -classpath .:.. tests.SM_JoinTest
Replacer: Clock



Any resemblance of persons in this database to people living or dead
is purely coincidental. The contents of this database do not reflect
the views of the University, the Computer  Sciences Department or the
developers...

**********************Query1 strating *********************
Query: Find the names of sailors who have reserved boat number 1.
     and print out the date of reservation.

  SELECT S.sname, R.date
  FROM   Sailors S, Reserves R
  WHERE  S.sid = R.sid AND R.bid = 1


(Tests FileScan, Projection, and Sort-Merge Join)
[Mike Carey, 05/10/95]
[David Dewitt, 05/11/95]
[Jeff Naughton, 05/12/95]

Query1 completed successfully!
******************Query1 finished!!!*****************



**********************Query3 strating *********************
```

Query: Find the names of sailors who have reserved a boat.

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
```

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]
[Mike Carey]
[Mike Carey]
[David Dewitt]
[David Dewitt]
[Jeff Naughton]
[Miron Livny]
[Yannis Ioannidis]
[Raghu Ramakrishnan]
[Raghu Ramakrishnan]

Query3 completed successfully!
**********Query3 finished!!!**********

**********Query4 strating **********
Query: Find the names of sailors who have reserved a boat
and print each name once.

```
SELECT DISTINCT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid
```

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]
[Jeff Naughton]
[Mike Carey]
[Miron Livny]
[Raghu Ramakrishnan]
[Yannis Ioannidis]

Query4 completed successfully!
**********Query4 finished!!!**********

**********Query5 strating **********
Query: Find the names of old sailors or sailors with a rating less
than 7, who have reserved a boat, (perhaps to increase the
amount they have to pay to make a reservation).

```
SELECT S.sname, S.rating, S.age
FROM   Sailors S, Reserves R
WHERE  S.sid = R.sid and (S.age > 40 || S.rating < 7)
```

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[Mike Carey, 9, 40.3]
[David Dewitt, 10, 47.2]
[David Dewitt, 10, 47.2]
[Jeff Naughton, 5, 35.0]
[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!
**********Query5 finished!!!**********

Finished joins testing
join tests completed successfully
make[1]: Leaving directory '/home/pi/Downloads/minjava/javaminibase/src/tests'
]0;pi@raspberrypi: ~/Downloads/minjava/javaminibase/src
[01;32mpi@raspberrypi[00m:[01;34m~/Downloads/minjava/javaminibase/src $[00m exit

Script done on 2022-02-06 16:15:56-07:00 [COMMAND_EXIT_CODE="0"]