1 The Assignment

For programming assignment 3 you should write a class that implements a B-tree of user-specified order. Your program should use the implementation that we discussed in class — not the implementation of B+-trees outlined in the text.

The keys in the B-tree will be positive ints less than $10^9$. You should implement the following operations:

- A one-parameter constructor. It’s argument, $m$, is the order of the B-tree.
- boolean member(int key): this will return true or false depending on whether key is in the B-tree.
- void insert(int key): this will insert the key into the B-tree as discussed in class.
- void makeEmpty(): this will “empty” the B-tree by making the root reference null.
- void print(): this should print a list of the nodes in the B-tree. Each node listing should show the keys in the node, and the children of the node. See below for a suggestion of how to indicate children.

Note that you are not required to implement remove.

The main method should begin by prompting the user for the order of the B-tree and creating an empty tree. It should then give the user a menu of single character commands and accept and carry out these commands until the user quits. The command menu should include single character commands for member, insert, makeEmpty, print, and quit. The member and insert commands should prompt for the key argument. Be sure your program output documents the meaning of the single character commands (e.g., (i)nsert.)

2 Classes

Your program should have a source file for the B-tree class. You may also want to have a source file for a B-tree node class. However, it’s probably easier to make the B-tree node class a nested class within the B-tree class. The main method can just be a public static method in the B-tree class file.

3 Errors

The input order of the B-tree will be an int and at least 3. If the user enters an invalid single-character command, your program should simply prompt for another. You can assume that the input values will be positive integers less than $10^9$. 
4 Implementation Issues

The lists of keys and the lists of child pointers in the nodes can either be linked lists or arrays. In order to somewhat simplify the code for searching a node, you might want to store a final “fake” key in each node. If the fake key is $10^9$, you can search for the correct branch (in \texttt{member}, \texttt{insert}, or \texttt{remove}) by looking for the first key greater than or equal to the argument.

Splitting of nodes will be easier if your nodes can store an extra key. For example, in a B-tree of order 4, the maximum number of keys that can be stored in a node is 3. However, it may be useful, prior to splitting, to allow 4 keys to be stored in a node.

By analogy with the binary search tree methods, it may be useful to implement private recursive methods corresponding to each of the public methods \texttt{member}, \texttt{insert}, and \texttt{print}. The recursive methods can take a pointer to a B-tree node as one of their arguments and traverse the tree by recursively calling themselves with child pointers.

In order to insure that everyone builds the same trees, you should split your nodes as follows:

1. Suppose that the keys in a node are indexed as if they were array elements. So the first key is key 0, the second key 1, etc. (Assume this regardless of whether you use an array or a linked list.)

2. When a node is split,
   
   (a) the keys in positions 0, 1, \ldots, \lceil m/2 \rceil - 1 become the new left child,
   
   (b) the key in position \( \lfloor m/2 \rfloor \) is moved “up,” and
   
   (c) the keys in positions \( \lfloor m/2 \rfloor + 1, \ldots, m - 1 \) become the new right child.

3. Here, \( m/2 \) is computed using the computer’s integer arithmetic. So in mathematical notation, it’s \( \lfloor m/2 \rfloor \). For example, \( 4/2 = 2 \) and \( 5/2 = 2 \).

4. As an example, if the B-tree has order 4, is initially empty, and the keys 1, 2, 3, 4 are inserted in order, then the new root should store the key 3, its new left-child should store the keys 1 and 2, and its new right-child should store the key 4.

It may be useful to write methods that return multiple values or objects. For example, the recursive insert method could return the middle key and the new right-child. If you want to do this, you can have the method return the right-child as the method value. The int can be returned by passing in an integer wrapper class that allows you to set the value of its int data member.

In the \texttt{print} method, you need to devise a method for indicating the children of a node. One approach is to assign each node a unique integer identifier when it’s created, and then simply print this identifier when you print each node, and list the identifiers of the children when you print the list of children. For example, if you build a B-tree of order 4 by inserting the keys 1, 2, 3, 4, in order, then as we noted before, the tree will have three nodes: the root will contain 3, its left child will contain 1 and 2, and its right child will contain 4. This could be printed as follows.

```
Id = 2, KeyCount = 1, ChildCount = 2
  K: 3 1000000000
  C: 0 1
Id = 0, KeyCount = 2, ChildCount = 0
  K: 1 2 1000000000
  C:
Id = 1, KeyCount = 1, ChildCount = 0
```
The identifiers can be generated by storing an int nextId data member. When a new node is created, the current value nextId is assigned to the new node, and nextId is then incremented. If the B-tree node class is a separate class, nextId can be a private static member of the class. If the B-tree node class is an inner class of the B-tree class, it can be an ordinary private data member of the B-tree class. (Inner classes can't have nonfinal static data members.)

5 Extra Credit

A correct and complete implementation of remove is worth 20 points. This method should use the algorithm we discussed in class.

6 Submitting your program

In order to submit your program you should copy all of your source files into a program3 subdirectory of your submit directory. You should create a program3 subdirectory by typing

```bash
mkdir /home/submit/cs245/<your userid>/program3
```

and then copying the source files to the submit directory. This can be done with

```bash
cp *.java /home/submit/cs245/<your userid>/program3
```

(You should replace <your userid> with your userid). You can check that the files were properly submitted by first changing your working directory to your submit directory with the command

```bash
cd /home/submit/cs245/<your userid>
```

You can check that you’re in the right directory by typing

```bash
pwd
```

(which stands for “print working directory”). This should print

```
/home/submit/cs245/<your userid>
```

You can list the files in your program 3 submit directory with the command

```bash
ls program3
```

If everything is OK, this should list all the files you wanted to submit. To make really sure you can look at the contents of these files with the less command. For example, you could see the contents of the submitted BTree.java by typing

```bash
less program3/BTree.java
```

The space bar will page forward through the file.
7 Due Date

In order to receive full credit, your program must be submitted electronically by 3:00 pm on Wednesday, April 6th, and you must turn in printouts of your source files by 6 pm on the 6th. You can receive half credit if you submit your program and turn in a print out after 3:00 pm Wednesday but before 3:00 pm Thursday.

8 Grading

Correctness will be 70% of your grade. Does your program correctly implement each of the required methods?

The following static features will be graded.

1. Documentation will be 10% of your grade. Does your header documentation include the author’s name, the purpose of the program, and a description of how to use the program? Are the identifiers meaningful? Are any obscure constructs clearly explained? Does the function header documentation explain the purpose of the function, its arguments, and its return value?

2. Source format will be 10% of your grade. Is the indentation and naming of identifiers consistent? Have blank lines been used so that the program is easy to read? Are there any lines of source that are longer than 80 characters?

3. Quality of solution will be 10% of your grade. Did you use the required class? Are any of your methods more than 40 lines? Are there long or multipurpose methods? Is your solution too clever — i.e., has the solution been condensed to the point where it’s incomprehensible?

9 Collaboration

It is OK for you to discuss solutions to this program with your classmates. However, no collaboration should ever involve looking at one of your classmate’s source programs! It is usually extremely easy to determine that someone has copied a program, even when the individual doing the copying has changed identifiers and comments. If we discover that someone has copied a program, the authors of both programs will receive an F on the assignment. Repeat offenders will receive an F in the course and may be referred to the academic honesty committee for additional disciplinary action.