Name: Answer Key

Email address: ____________________

CSE 373 Sample Midterm #2

(closed book, closed notes, calculators o.k.)

Instructions Read the directions for each question carefully before answering. We will give partial credit based on the work you **write down**, so show your work! Use only the data structures and algorithms we have discussed in class or which were mentioned in the book so far.

Note: For questions where you are drawing pictures, please circle your final answer for any credit. There is one extra page at the end of the exam that you may use for extra space on any problem. If you detach this page it must still be turned in with your exam when you leave.

Advice You have 50 minutes, **do the easy questions first**, and work quickly!

1) **True/False**. Circle True or False below. You do *not* need to justify your answers.

a Linear probing is equivalent to double hashing with a secondary hash function of $h_2(k) = 1$. **True** **False**

b. If $T_1(N) = O(f(n))$ and $T_2(N) = O(f(n))$, then $T_1(N) = O(T_2(N))$. **True** **False**

c. Building a heap from an array of N items requires $\Omega(N \log N)$ time. **True** **False**

d. Merging heaps is faster with binary heaps than with leftist or skew heaps, because we only need to concatenate the heap arrays and then run BuildHeap on the result. **True** **False**

2) **Short Answer Problems:** *Be sure to answer all parts of the question!!*

a) Which ADT do binomial queues implement? If we forget simplicity of implementation, are they better than binary heaps? Are they better than leftist heaps? Justify your answer.

- Priority Queue
- Yes – BinQ can do insert, deletemin, + merge in worst case $O(\log N)$
  (Bin Heaps can do insert + deletemin in $O(\log N)$ but merge takes $O(N)$)
- Yes – Insert will be faster (on average) w. BinQ

b) For node i in a ThreeHeap, give formulas that calculate each of the following:
   (1) Node i's parent
   (2) Node i's three children

(Assuming the first element is placed in location 1)

(1) Parent = $\lfloor (i+1)/3 \rfloor$

(2) Children =

$(i*3)-1, \quad i*3, \quad i*3+1$

c) What is the main advantage that open addressing hashing techniques have over separate chaining?

- no extra space for pointers
- no extra time to allocate nodes/structures that make up each bucket
- spatial locality when re-hashing (reading the original hash table)

d) Fill in the blank in the following text to make the statement true.

In the union/find data structure of N items, if we use union-by-size *without path compression*, then any combination of M union and/or find operations takes at most __________________ time.

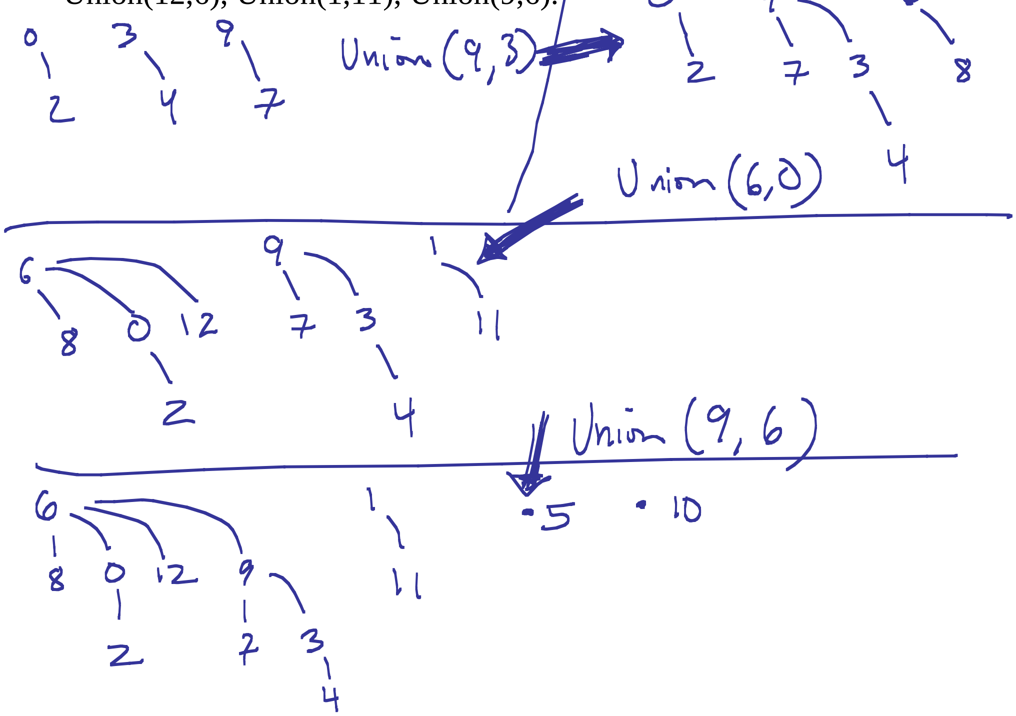
$O(M \log N)$

3) **Disjoint Sets:**

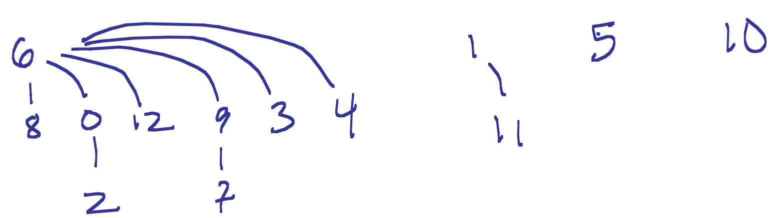
Consider the set of initially unrelated elements 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

a.) Draw the final forest of up-trees that results from the following sequence of operations using on union-by-size. Break ties by keeping the first argument as the root.

Union(0,2), Union(3,4), Union(9,7), Union(9,3), Union (6,8), Union(6,0), Union(12,6), Union(1,11), Union(9,6).



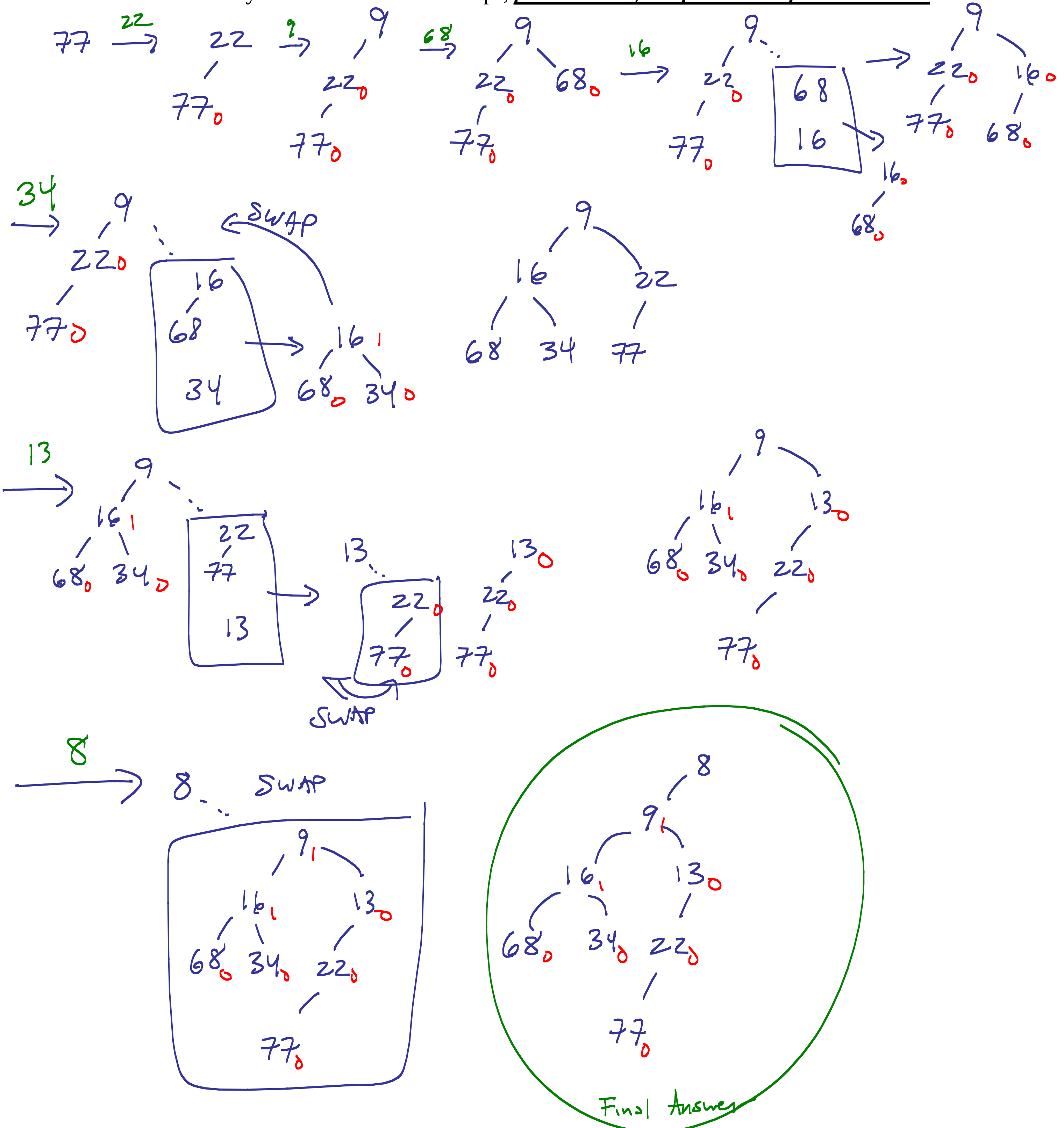
b.) Draw the new forest of up-trees that results from doing a Find(4) with path compression on your forest of up-trees from (a).



(NPL in Red)

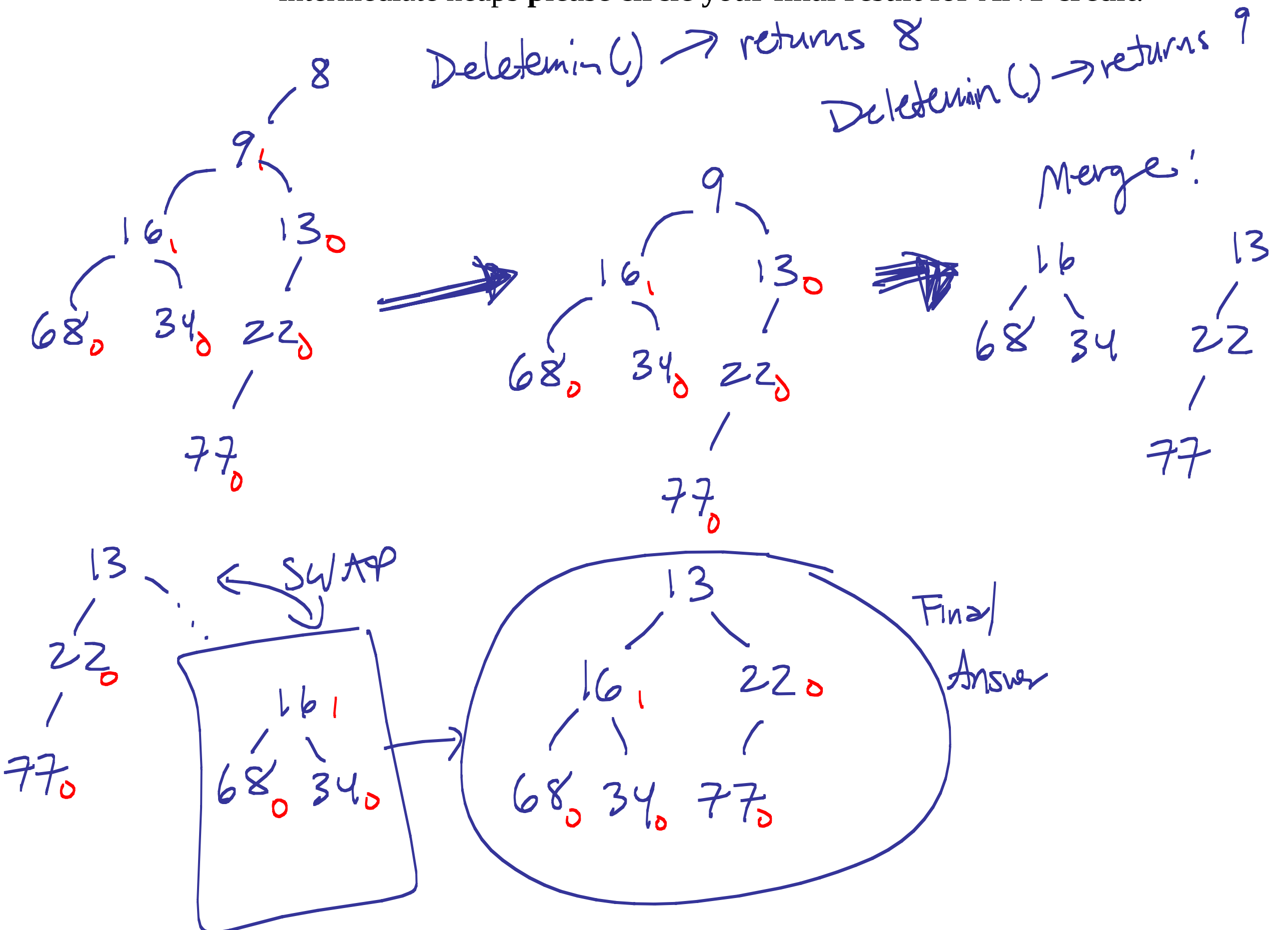
4) **Heaps**

a) Draw the leftist heap that results from inserting: 77, 22, 9, 68, 16, 34, 13, 8 in that order into an initially empty leftist heap. You do not need to show the array representation of the heap. You are only required to show the final heap, although if you draw intermediate heaps, ***please circle your final result for ANY credit.***



4 (cont.) **Heaps:**

b) Draw the leftist heap that results from doing 2 deletemins on the heap you created in part a). You are only required to show the final heap, although if you draw intermediate heaps **please circle your final result for ANY credit**.



c) What is the null path length of the root node in the last heap you drew in part b) above?

one

d) List 2 good reasons why you might choose to use a skew heap rather than a leftist heap.

- easier to implement / don't need to store or calculate null path length
- if you don't care about worst case time of individual operations, but of a set of operations (Amortized time bound is o.k.)

5) **Binomial Queues** –

a) What is the minimum and maximum number of nodes in a Binomial ***Tree*** of height h?

$min = 2^h$

$max = 2^h$

b) What is the minimum and maximum number of nodes in a Binomial ***Queue*** whose tallest tree is of height h?

$min = 2^h$

$max = 2^{h+1} - 1$

c) Briefly describe how `Findmin()` is implemented for Binomial Queues.

The smallest value in each tree will be stored at its root. So to find the minimum value in the entire queue, just examine each root + determine the smallest root.

6) Draw the contents of the hash table in the boxes below given the following conditions:

The size of the hash table is 12.
Open addressing and double hashing is used to resolve collisions.
The hash function used is H(k) = k mod 12
The second hash function is: H2(k) = 7 – (k mod 7)

What values will be in the hash table after the following sequence of insertions? Draw the values in the boxes below, and show your work for partial credit.

33, 10, 9, 13, 12, 45, 26, 17

	0	12	
$17_2 \rightarrow$	1	13	$\leftarrow 45_1$
	2	9_1	$\leftarrow 26_0$
	3		
	4	26_1	
$17_3 \rightarrow$	5	45_2	$\leftarrow 17_0$
	6		
	7		
	8		
$17_1 \rightarrow$	9	33	$\leftarrow 9_0$, 45_0
	10	10	
	11		

33 mod 12 = 9

10 mod 12 = 10

9 mod 12 = 9

$H2(9) = 7 - (9 \bmod 7) = 7 - 2 = 5$

13 mod 12 = 1

12 mod 12 = 0

45 mod 12 = 9

$H2(45) = 7 - (45 \bmod 7) = 7 - 3 = 4$

26 mod 12 = 2

$H2(26) = 7 - (26 \bmod 7) = 7 - 5 = 2$

17 mod 12 = 5

$H2(17) = 7 - (17 \bmod 7) = 7 - 3 = 4$

Cannot find a spot for 17. (Maybe 12 is not a good choice of table size?)

In general you don't write your programs to optimize for locality of instructions, instead you would optimize for locality on data. But locality still occurs on instructions.

7) Memory

a) Define spatial locality.

(posted later)

b) Give an example of *spatial* locality when accessing **instructions** in your program. Give a short example (using code) and indicate what will have spatial locality and why.

Accessing sequential instructions with no branches (loops, if's, function calls)

ex.

```
x = y + 1;
a = b / c + 36;
d = e * f;
```

instructions would be executed in order. Would also be located next to each other in memory. If reading the first instruction was a miss, reading the second two may be cache hits.

c) Define temporal locality.

(posted later)

d) Give an example of *temporal* locality when accessing **instructions** in your program. Give a short example (using code) and indicate what will have temporal locality and why.

```
for (int i=0; i<MAX; i++){
    a = b + c + 362;
}
```

these instructions will be executed MAX times in a row, all very close in time, thus are likely to remain in cache.

Be sure the values you pick for L+M will not exceed the page size. (round down if needed)

8) B-trees

page size

$$L = \frac{1024}{128} = 8$$

data

a) Given the following parameters:

- Disk access time = 1milli-sec per byte
- 1 Page on disk = 1024 bytes
- Key = 16 bytes
- Pointer = 4 bytes
- Data = 128 bytes per record (includes key)

What are the best values for:

M = 52

L = 8

$$\underbrace{4}_{\text{pointer size}} \cdot M + \underbrace{16}_{\text{key size}}(M-1) = \underbrace{1024}_{\text{Page size}}$$

(Solve for M)

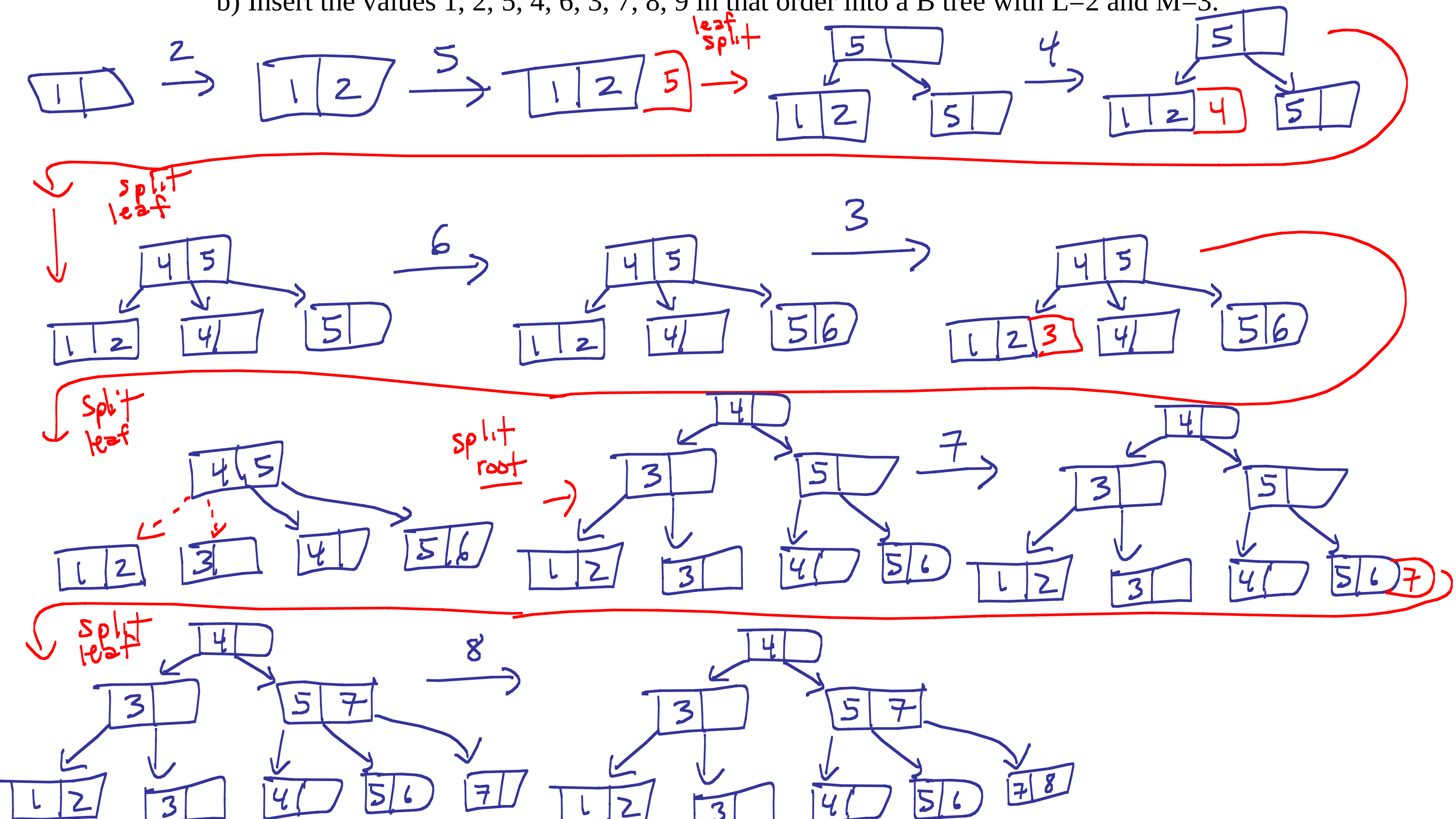
$$4M + 16M - 16 = 1024$$

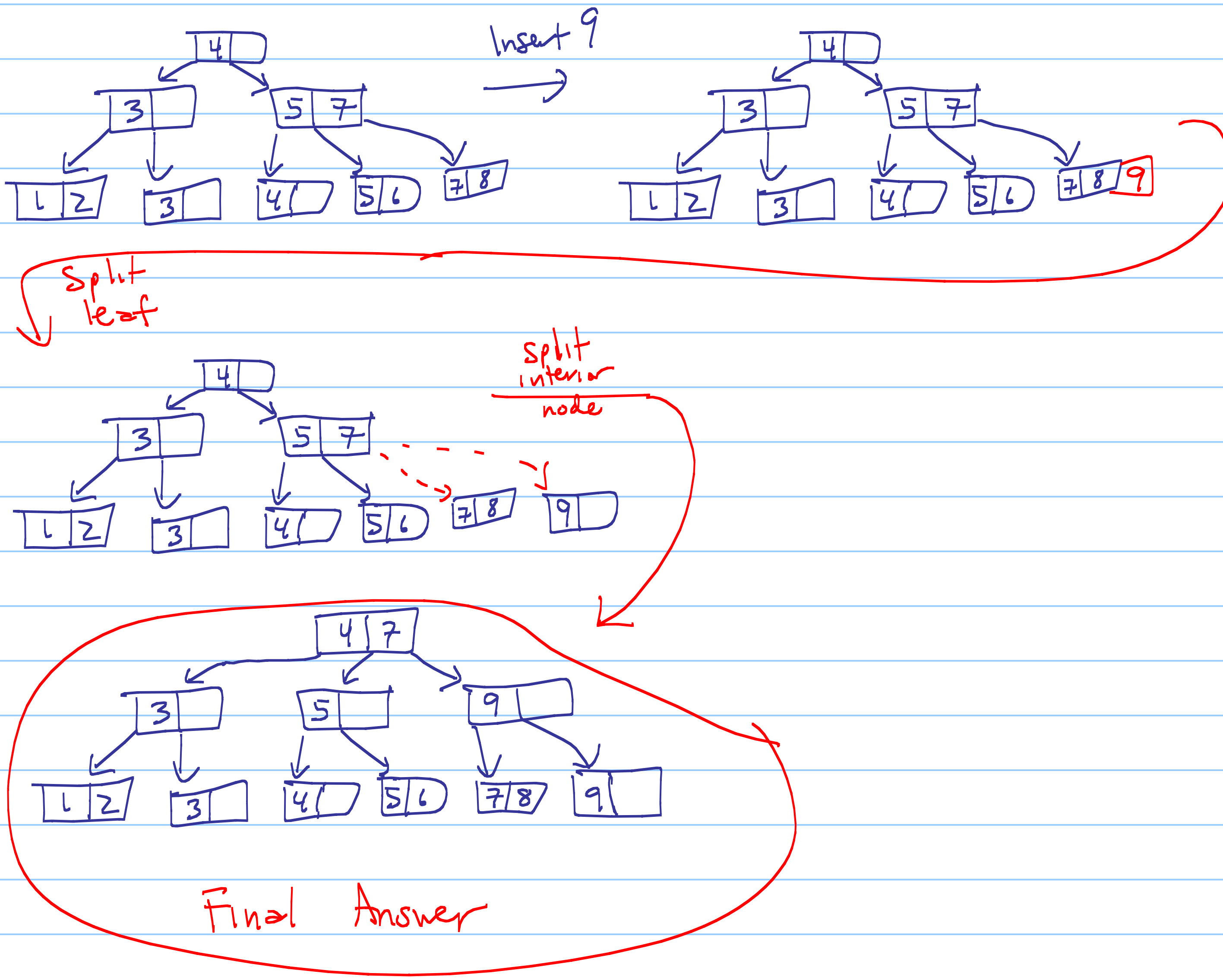
$$20M = 1040$$

$$M = 52$$

L=2 M=3

b) Insert the values 1, 2, 5, 4, 6, 3, 7, 8, 9 in that order into a B tree with L=2 and M=3.





Insert 9
Split leaf
Split interior node
Final Answer