CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2020)
PROF. ANDY PAVLO

Homework #2 (by Ian Romines)
Due: **Sunday October 4, 2020 @ 11:59pm**

IMPORTANT:

- **Upload this PDF** with your answers to **Gradescope by 11:59pm on Sunday October 4, 2020.**
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.
- **You have to use this PDF for all of your answers.**

For your information:

- Graded out of **100** points; **4** questions total
- Rough time estimate: ≈1-4 hours (0.5-1 hours for each question)

Revision : 2020/09/29 10:44

Question	Points	Score
Cuckoo Hashing	20	
B+Tree	45	
Extendible Hashing	25	
Suffix Trees	10	
Total:	100	

Question 1: Cuckoo Hashing . . . . . [20 points]

Consider the following cuckoo hashing schema:

1. Both tables have a size of 4.
2. The hashing function of the first table returns the lowest two bits: $h_1(\texttt{x}) = \texttt{x \& 0b11}$.
3. The hashing function of the second table returns the next two bits: $h_2(\texttt{x}) = \texttt{(x >> 2) \& 0b11}$
4. When replacement is necessary, first select an element in the second table.
5. The original content is shown in Figure 1.

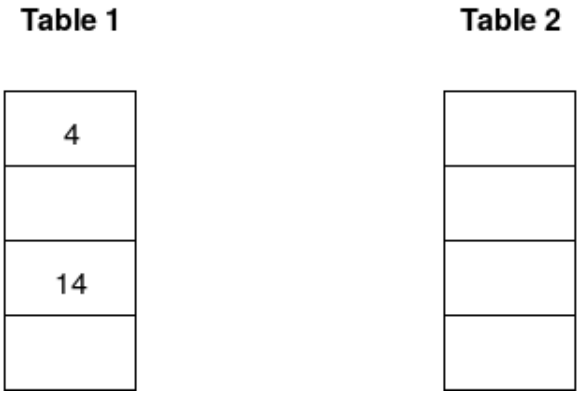
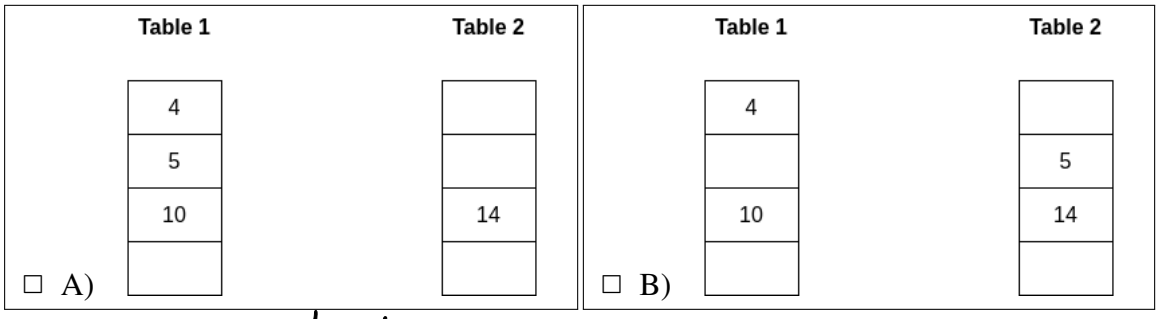


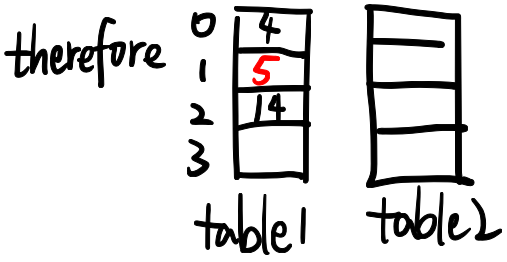
Figure 1: Initial contents of the hash tables.

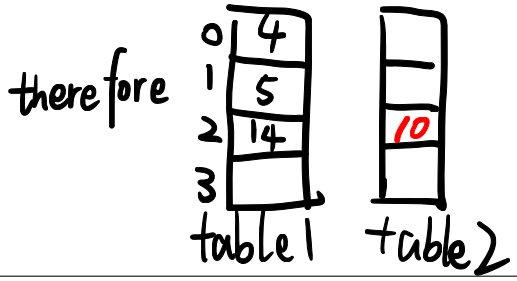
(a) **[4 points]** Insert keys 5 and 10. Select the resulting two tables. C

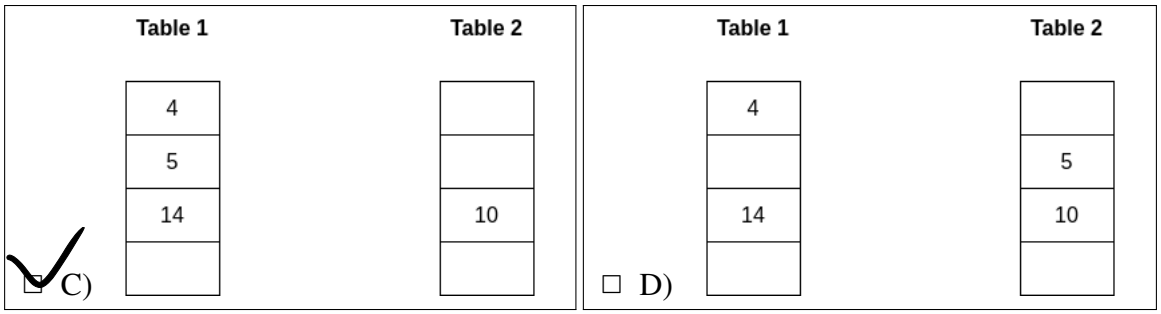


① insert 5, $hash_1(5)=1$

② insert 10, $hash_1(10)=2$ occupied
thus go table 2, $hash_2(10)=2$

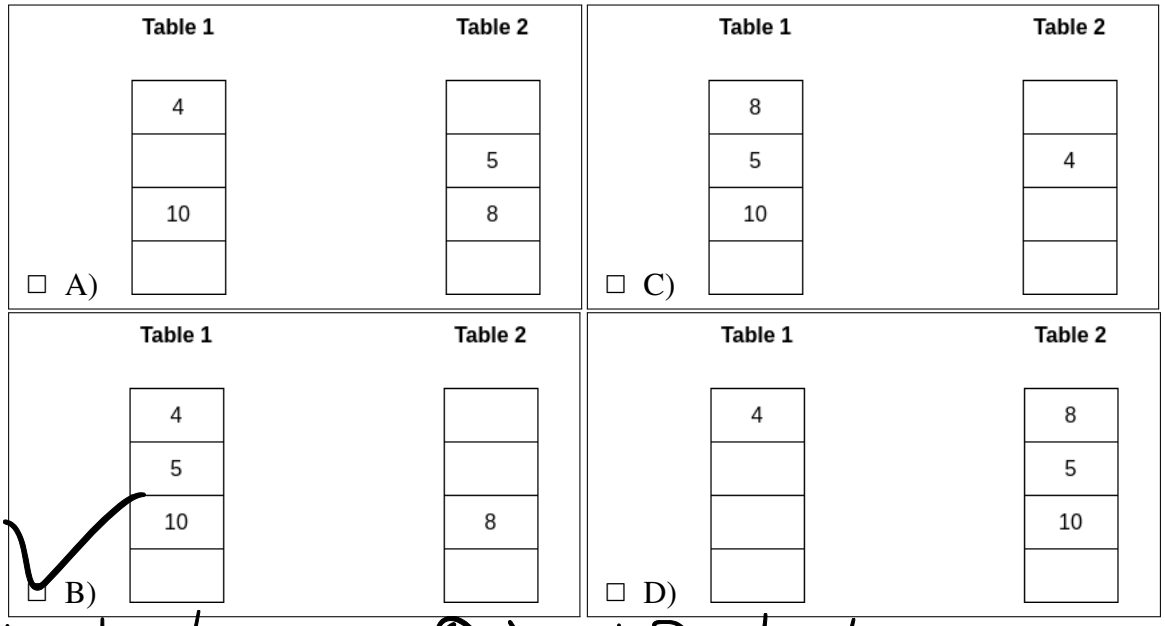






0b1110 0b1000

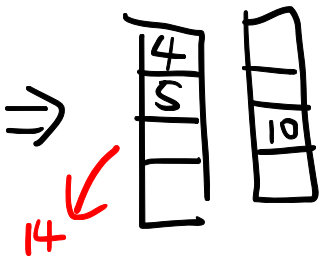
(b) **[4 points]** Then delete 14, and insert 8. Select the resulting two tables.

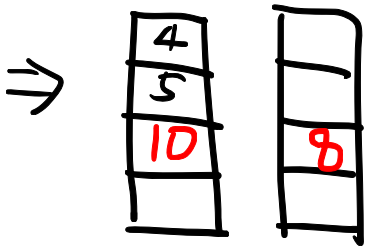


① delete 14 : $hash_1(14) = 2$ ② insert 8 $hash_1(8) = 0$ x occupied

$hash_2(8) = 2$ x occupied

take 10 : $hash_1(10) = 2$, empty ✓



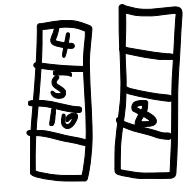


therefore choose B

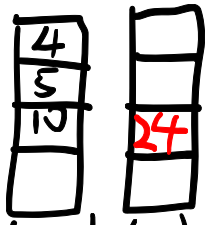
4
5
10 24

(c) **[6 points]** Finally, insert 24. Select the resulting two tables.

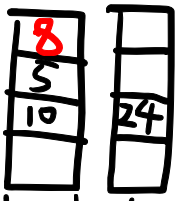
0b 0001 1001



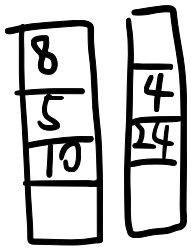
hash$_1$(24) = 0
hash$_2$(24) = 2



hash$_1$(8) = 0



hash$_2$(4) = 1



☐ A)

Table 1	Table 2
8	
5	4
10	

☐ C)

Table 1	Table 2
4	24
	5
8	10

☑ B)

Table 1	Table 2
8	
5	4
10	24

☐ D)

Table 1	Table 2
8	
	4
10	24

(d) **[6 points]** What is the smallest key that potentially causes an infinite loop given the tables in **(c)**

☐ 0 ☐ 2 ☐ 3 ☑ 6 ☐ 7 ☐ 9 ☐ None of the above

$h_1(0) = 0$ ✗ $h_2(0) = 0$ ✓

$h_1(2) = 2$ ✗ $h_2(2) = 0$ ✓

$h_1(3) = 3$ ✗ $h_2(3) = 0$ ✓

$h_1(6) = 2$ ✗ $h_2(6) = 1$ ✗

Question 2: B+Tree . . . . . [45 points]

Consider the following B+tree.

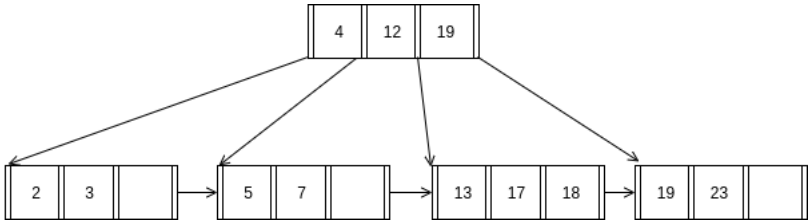
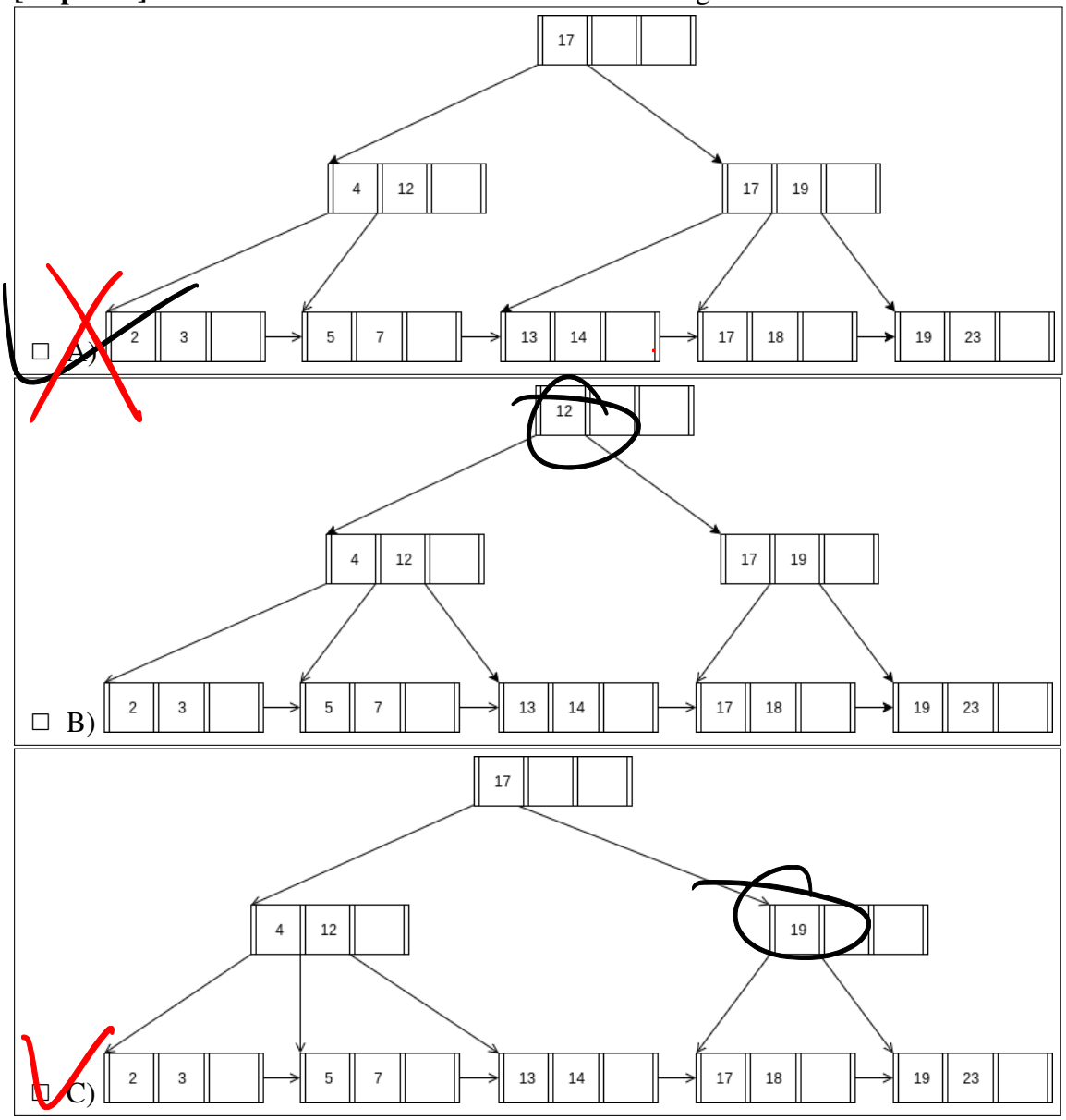


Figure 2: B+ Tree of order $d = 4$ and height $h = 2$.

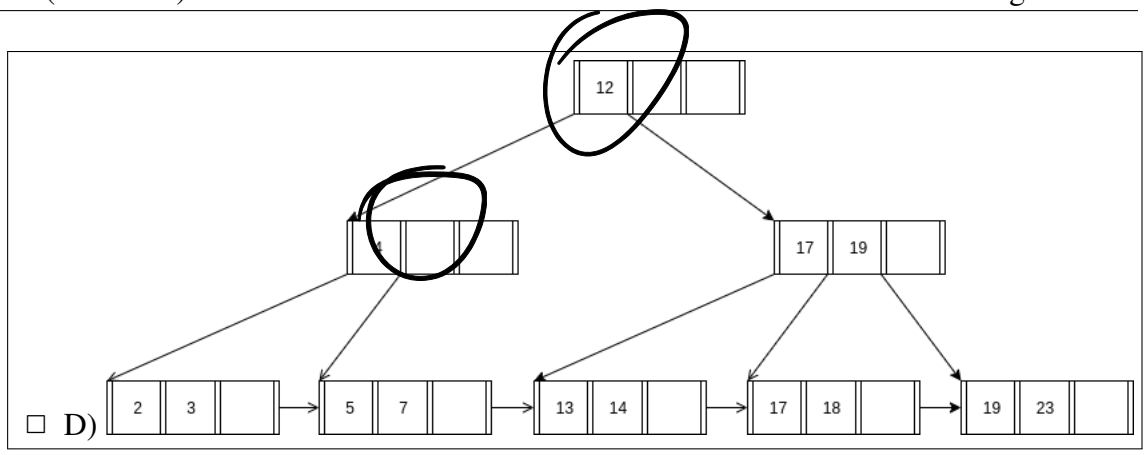
When answering the following questions, be sure to follow the procedures described in class and in your textbook. You can make the following assumptions:

- A left pointer in an internal node guides towards keys $<$ than its corresponding key, while a right pointer guides towards keys $\geq$.
- A leaf node underflows when the number of **keys** goes below $\lceil \frac{d-1}{2} \rceil$.
- An internal node underflows when the number of **pointers** goes below $\lceil \frac{d}{2} \rceil$.

(a) **[15 points]** Insert 14^* into the B+tree. Select the resulting tree.



☐ D)

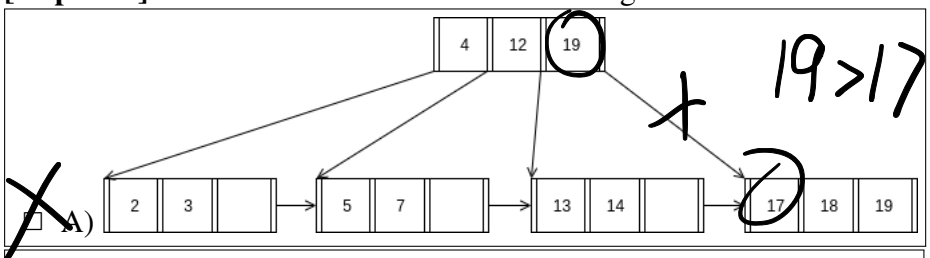


(b) **[5 points]** How many pointers (parent-to-child and sibling-to-sibling) do you chase to find all keys between 5 and 15?

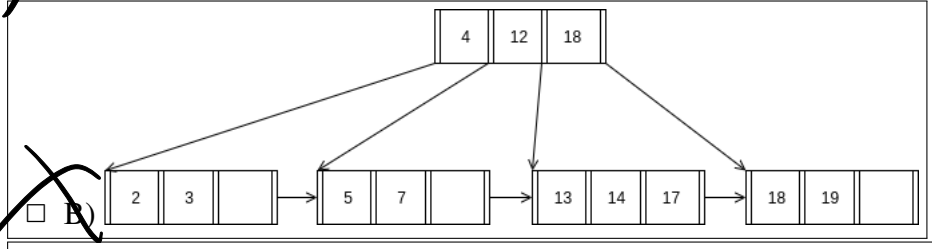
☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

(c) **[15 points]** Then delete 23^*. Select the resulting tree.

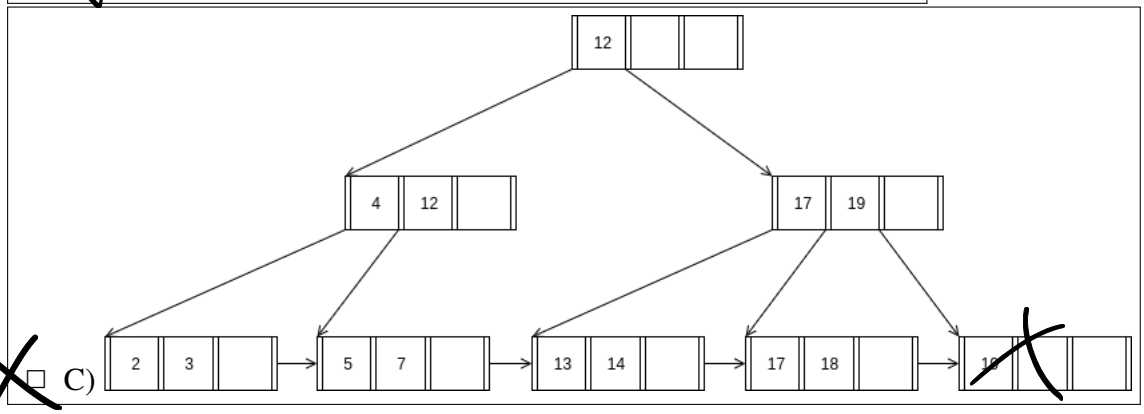
☐ A)

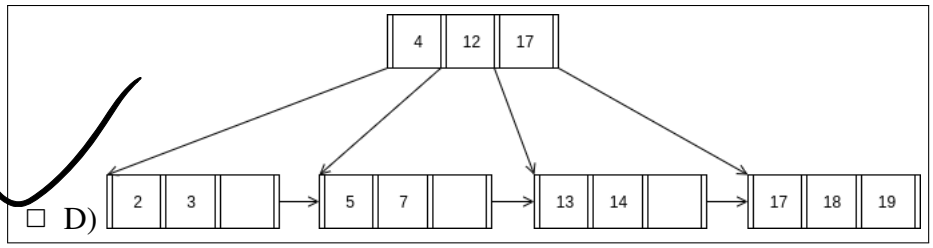


☐ B)



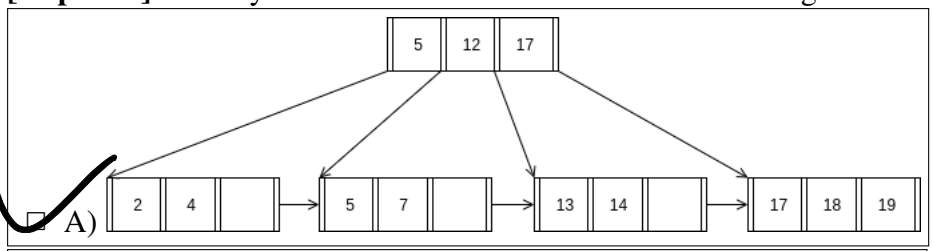
☐ C)



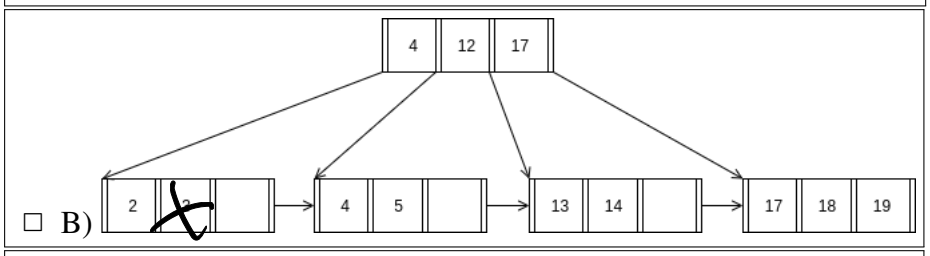


☐ D)

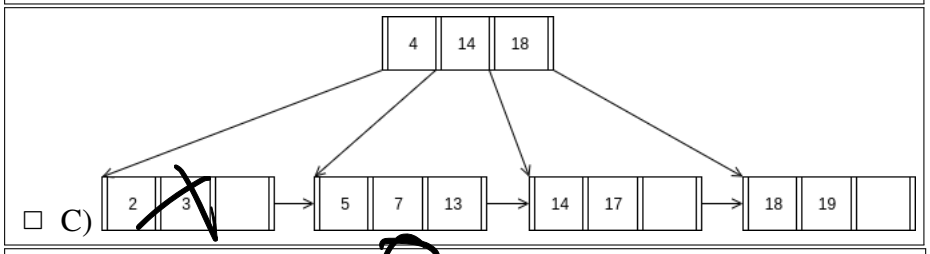
(d) **[10 points]** Finally insert 4^* and delete 3^*. Select the resulting tree.



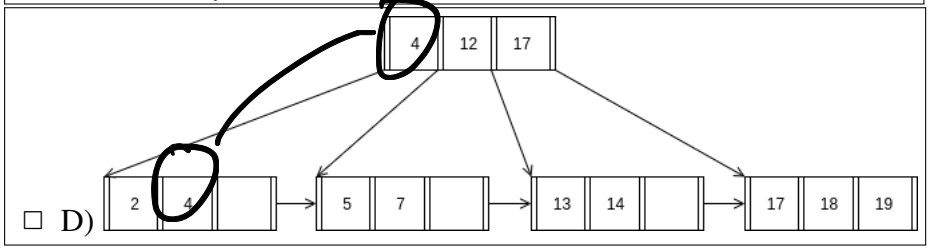
☐ A)



☐ B)



☐ C)



☐ D)

Question 3: Extendible Hashing . . . . . [25 points]

Consider an extendible hashing structure such that:

- Each bucket can hold up to two records.
- The hashing function uses the lowest g bits, where g is the global depth.

(a) Starting from an empty table, insert keys 15, 2, 31, 11.

i. **[3 points]** What is the global depth of the resulting table?
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ None of the above

ii. **[3 points]** What is the local depth the bucket containing 2?
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ None of the above

iii. **[3 points]** What is the local depth of the bucket containing 31?
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ None of the above

(b) Starting from the result in **(a)**, you insert keys 6, 9, 23, 12, 11.

i. **[4 points]** Which key will first cause a split (without doubling the size of the table)?
☐ 6 ☐ 9 ☐ 23 ☐ 12 ☐ 11 ☐ None of the above

ii. **[4 points]** Which key will first make the table double in size?
☐ 6 ☐ 9 ☐ 23 ☐ 12 ☐ 11 ☐ None of the above

(c) Now consider the table below, along with the following deletion rules:

1. If two buckets have the same local depth d, and share the first $d-1$ bits of their indexes (e.g. `010` and `110` share the first 2 bits), then they can be merged if the total capacity fits in a single bucket. The resulting local depth is $d-1$.
2. If the global depth g becomes strictly greater than all local depths, then the table can be halved in size. The resulting global depth is $g-1$.

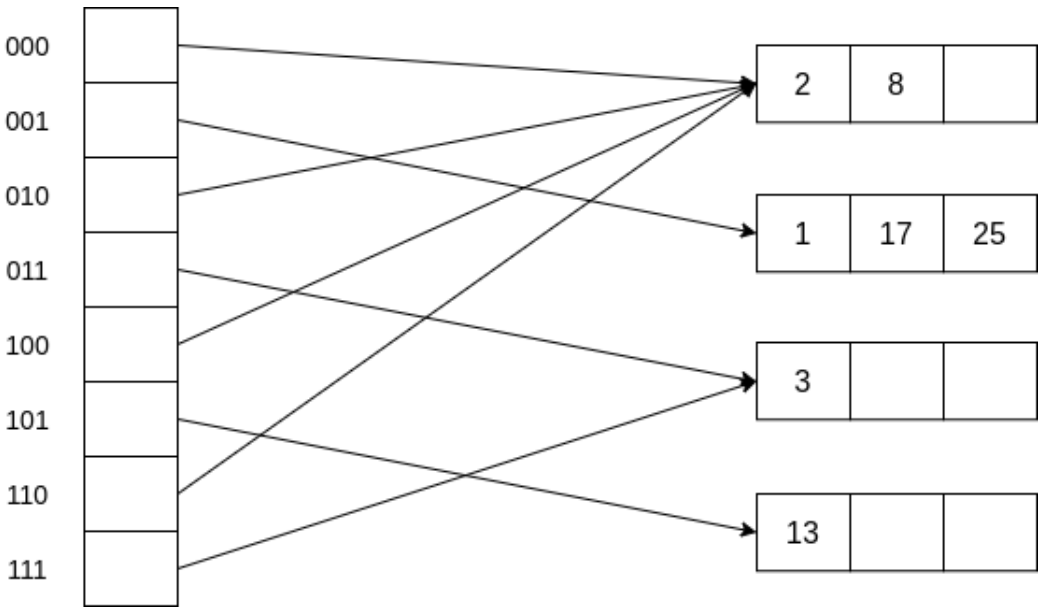


Figure 3: Extendible Hash Table along with the indexes of each bucket

Starting from the table above, delete keys 3, 8, 1, 2, 17.

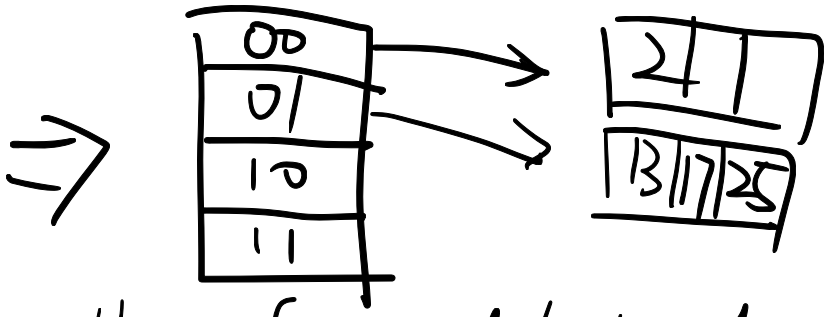
i. **[4 points]** Which deletion first causes a reduction in a local depth.
☐ 3 ☐ 8 ☑ 1 ☐ 2 ☐ 17 ☐ None of the above

ii. **[4 points]** Which deletion first causes a reduction in global depth.
☐ 3 ☐ 8 ☑ 1 ☐ 2 ☐ 17 ☐ None of the above

delete 3 can't merge

delete 8 can't merge

delete 1 merge { 001
101



therefore delete 1 would cause reduction

Question 4: Suffix Trees ..... [10 points]

Consider the following suffix tree for **unsigned 32-bit integers**.

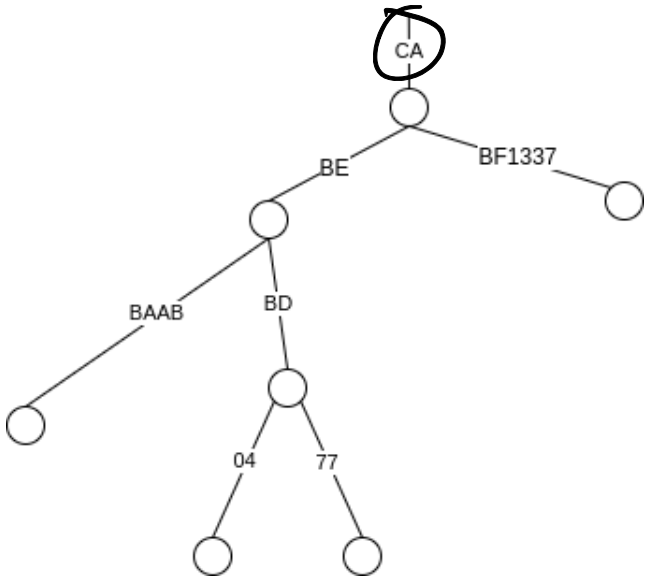


Figure 4: Suffix Tree

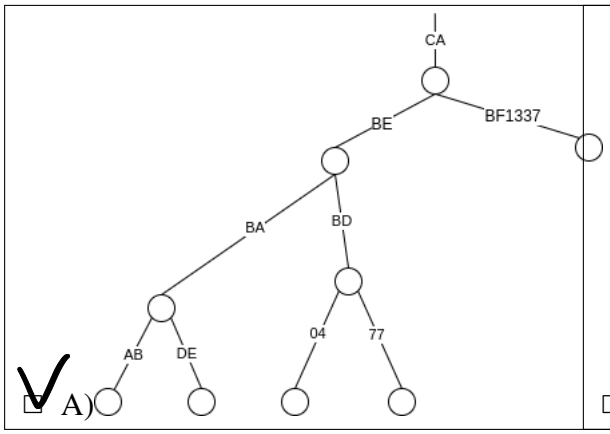
starts with CA

(a) **[3 points]** Which of the following elements belong to the suffix tree. Select all that apply.

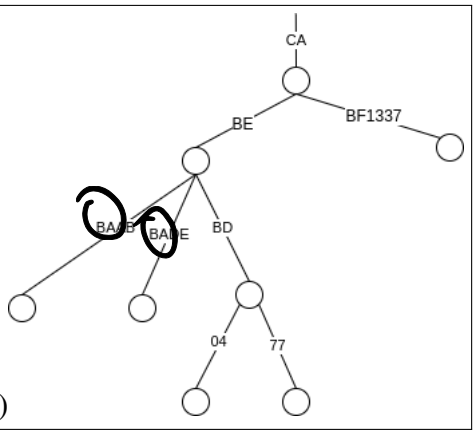
☐ 0xCABEACCA ☐ 0xCA1337BF ☐ 0x77BDBECA ☐ 0xBAABCABE ☐ 0xCABEBC04

☐ None of the above

(b) **[7 points]** Insert the key 0xCABEBADE. Select the resulting tree.

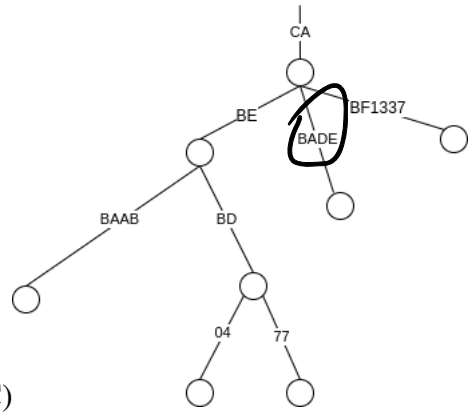


☐ A)



☐ B)

☐ C)



☐ D)