MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

DEPARTMENT
OF COMPUTER ENGINEERING

CNG 351
Data Management and File Structures
Assignment 5

Team Details

Member 1

Name: Mert Can
Surname: Bilgin
Student ID: 2453025

Member 2

Name: Hami
Surname: Karslı
Student ID: 2453322

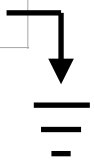
Question 1

a) The data is started to insert in order that starting from 501. In blocks, P represents the pointer.

h(subscription_id) = subscription_id mod 5, so h(501) = 501 mod 5 = 1. It means the data will be stored in Block 1.

Block 1

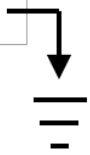
subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
501	high	450	credit-card	20/01/2018	



Then, h(502) = 502 mod 5 = 2. The data will be stored in Block 2.

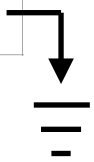
Block 1

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit-card	20/01/2018	



Block 2

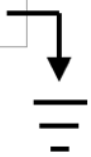
subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
502	medium	350	paypal	11/12/2022	



h(503) = 503 mod 5 = 3, so the data will be stored in Block 3.

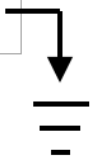
Block 1

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit-card	20/01/2018	



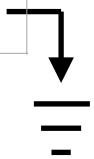
Block 2

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
502	medium	350	paypal	11/12/2022	



Block 3

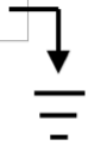
subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
503	low	150	bank-transfer	30/8/2018	



h(504) = 504 mod 5 = 4, so the data will be stored in Block 4.

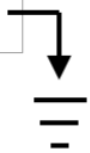
Block 1

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit- card	20/01/2018	



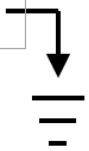
Block 2

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
502	medium	350	paypal	11/12/2022	



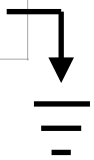
Block 3

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
503	low	150	bank- transfer	30/8/2018	



Block 4

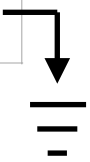
subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
504	high	450	credit- card	3/1/2020	



h(505) = 505 mod 5 = 0, so the data will be stored in Block 0.

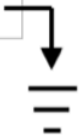
Block 0

subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
505	low	450	credit-card	15/10/2022	



Block 1

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit-card	20/01/2018	

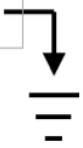


Block 2

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
502	medium	350	paypal	11/12/2022	

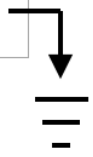
Block 3

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
503	low	150	bank-transfer	30/8/2018	

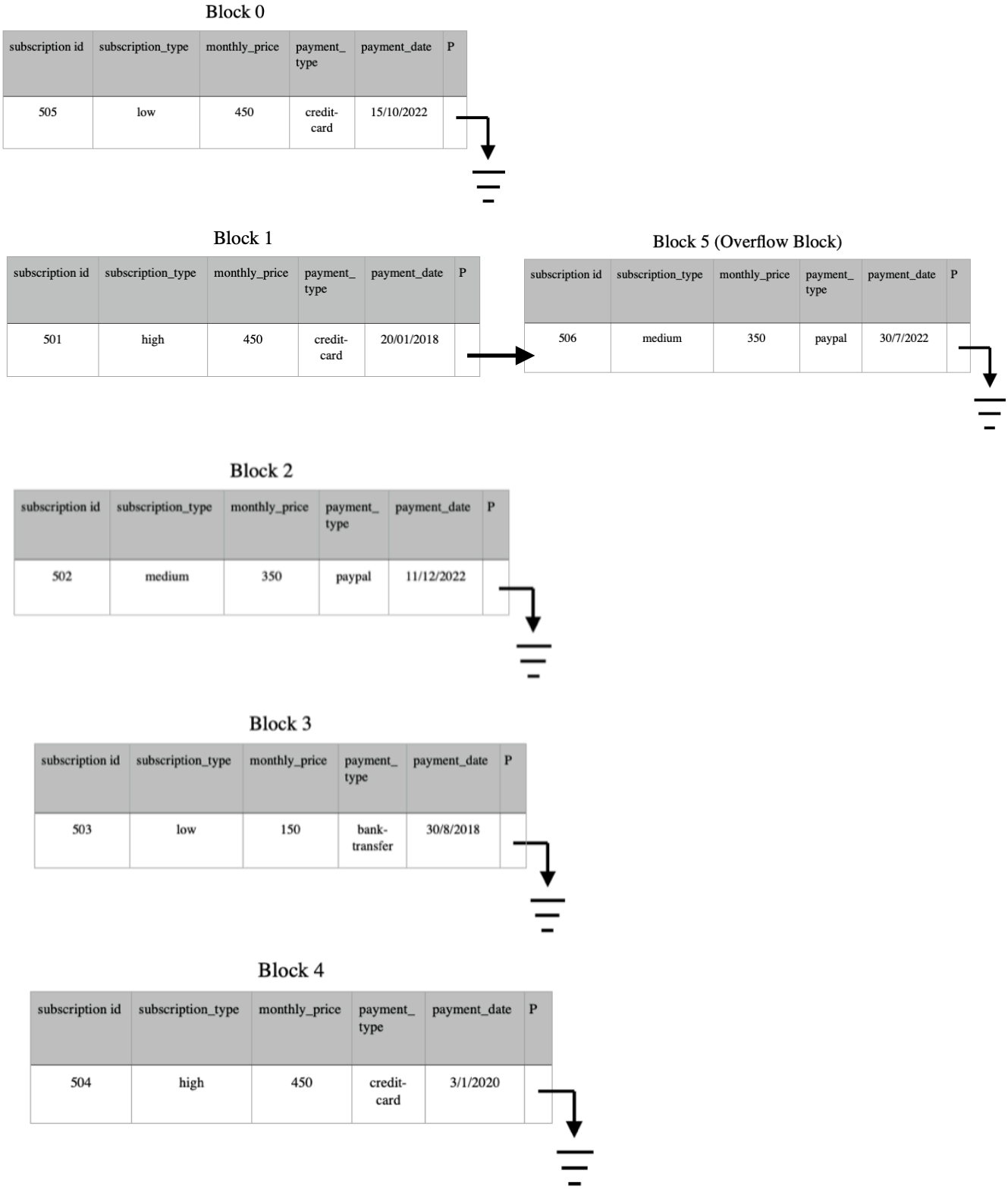


Block 4

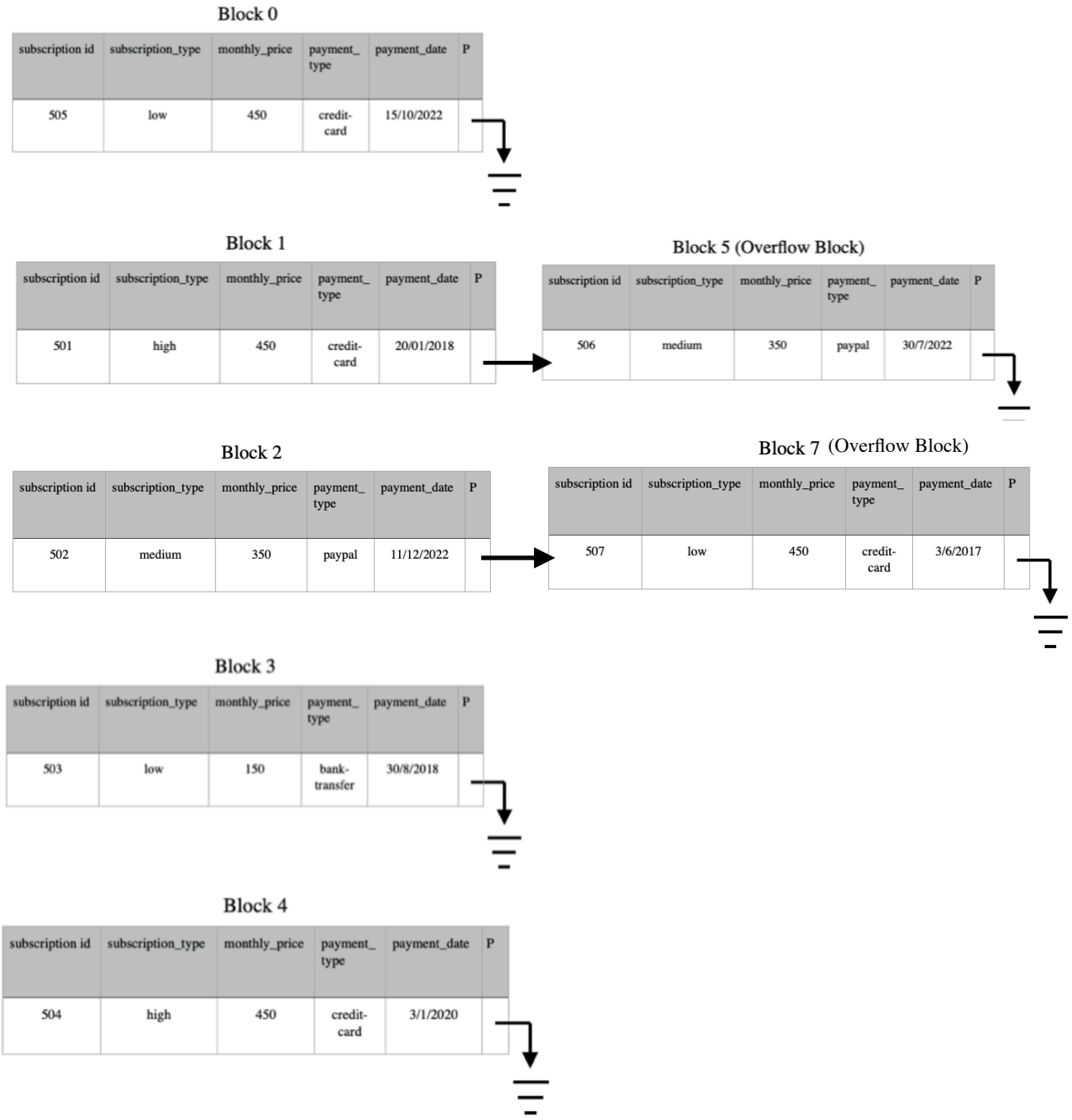
subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
504	high	450	credit-card	3/1/2020	



h(506) = 506 mod 5 = 1, the should be stored in Block 1, but there is a collision. Because we use chained overflow approach, it will stored in an overflow block.



h(507) = 507 mod 5 = 2, the data should be stored in Block 2, but there is a collision. Because we use chained overflow approach, it will stored in an overflow block.

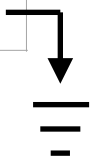


b) Again the data will be inserted in order starting from 501.

h (501 + 2018) mod 5 = 4, so the data will be inserted in B4.

B4

subscription id	subscription_ type	monthly_p rice	payme nt_type	payment_ date	P
501	high	450	credit-card	20/01/2018	



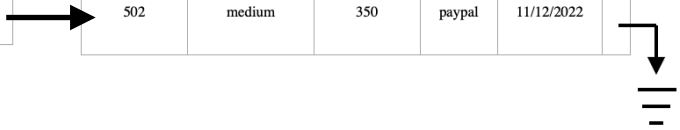
h (502 + 2022) mod 5 = 4. In this case there is a collision, so the data will be stored in an overflow block.

B4

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit-card	20/01/2018	

B5

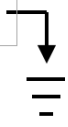
subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
502	medium	350	paypal	11/12/2022	



h (503 + 2018) mod 5 = 1, so the data will be stored in B1.

B1

subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
503	low	150	bank-transfer	30/8/2018	

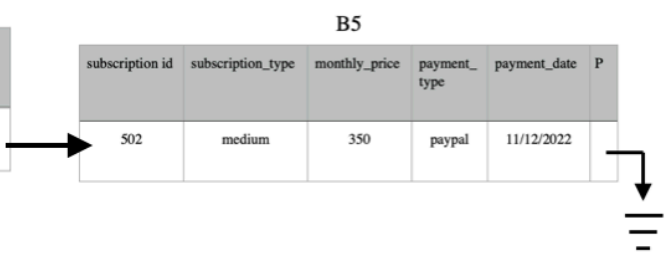


B4

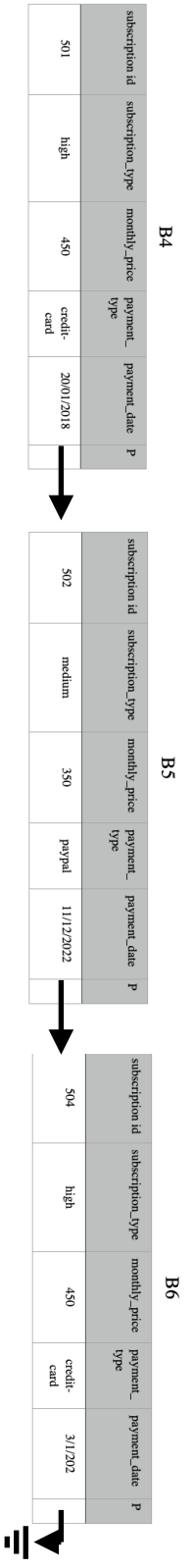
subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
501	high	450	credit-card	20/01/2018	

B5

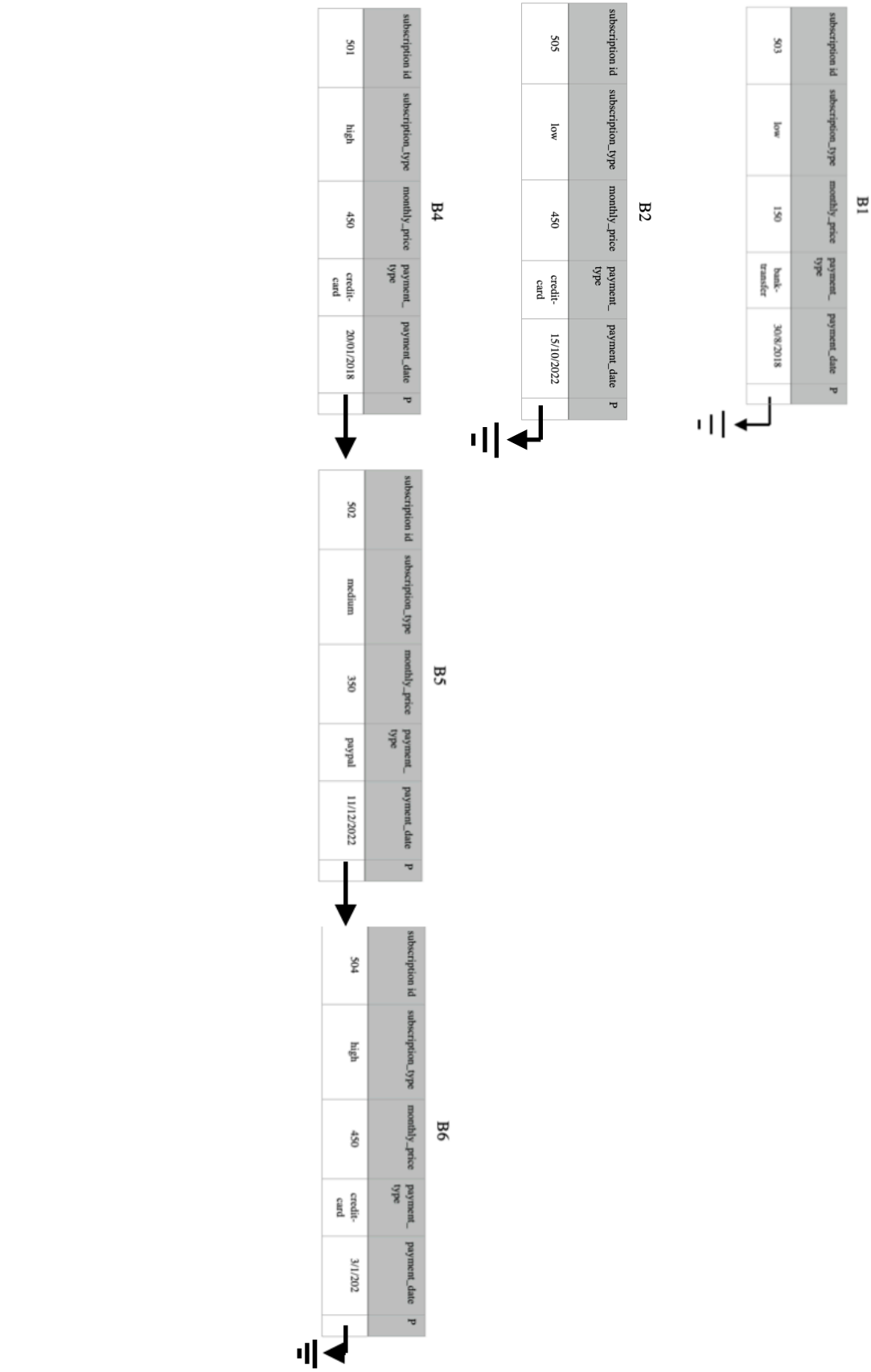
subscription id	subscription_type	monthly_price	payment_ type	payment_date	P
502	medium	350	paypal	11/12/2022	



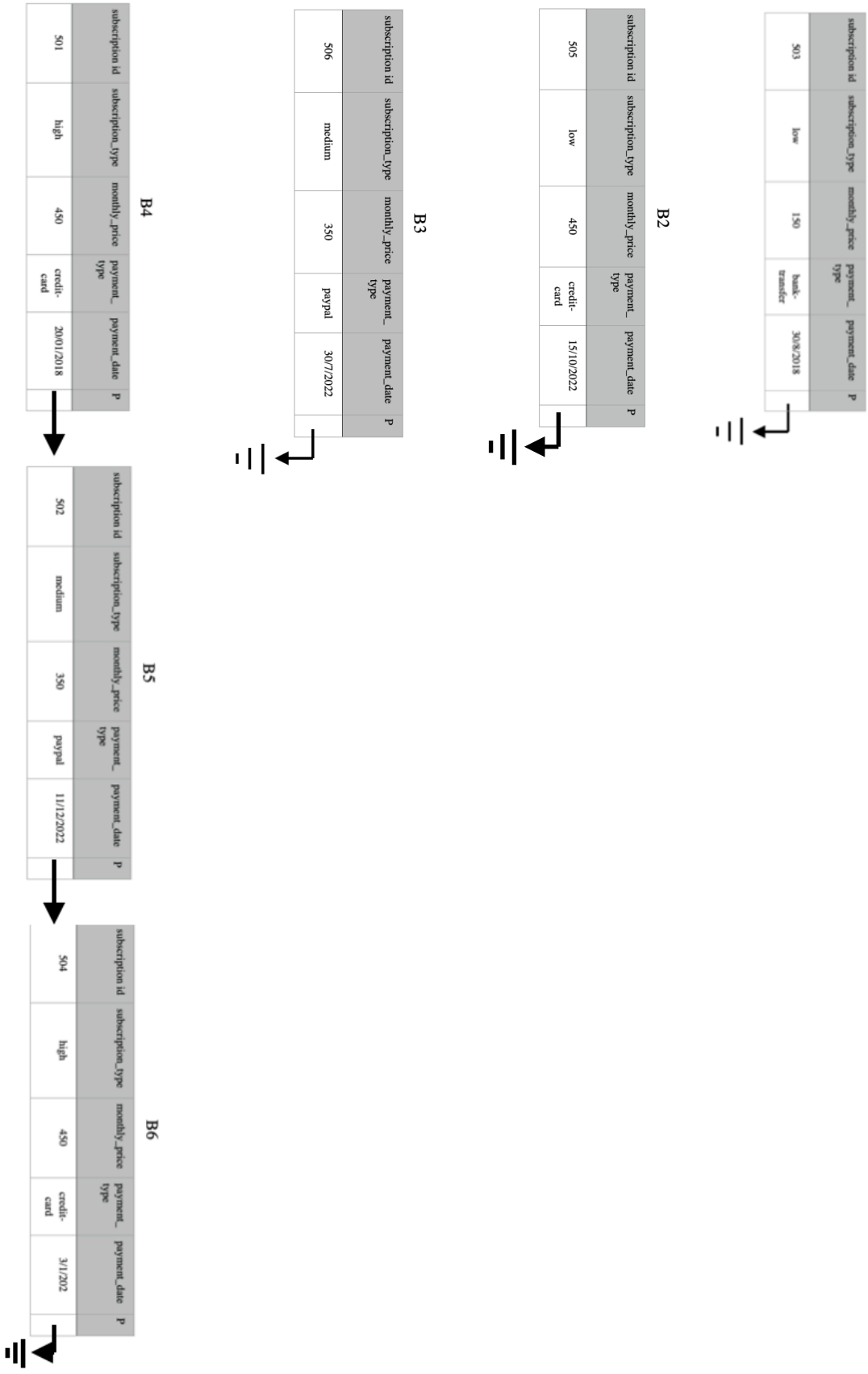
h (504 + 2020) mod 5 = 4. Again there is a collision, the data will be stored in another overflow block because one record can fit in one block.



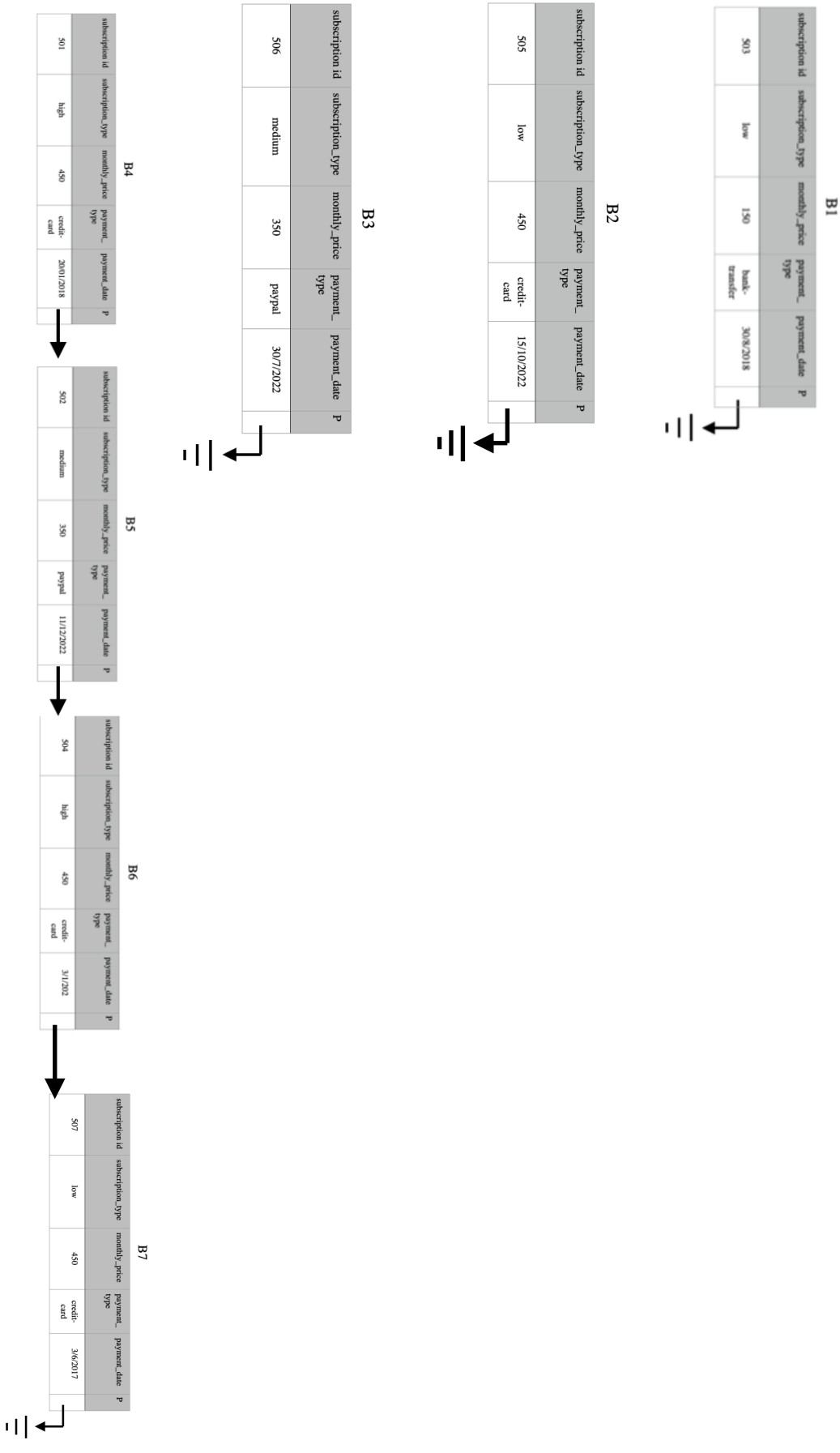
h (505 + 2022) mod 5 = 2, so the data will be stored in B2.



h (506 + 2022) mod 5 = 3, so the data will be stored in B3.



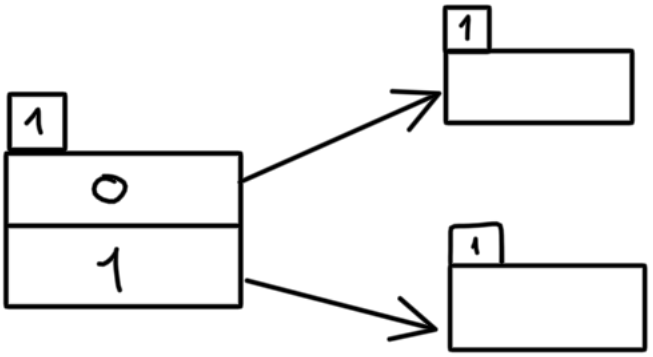
h (507 + 2017) mod 5 = 4. There is a collision, so the data will be stored in an overflow block.



c) I would prefer to use h(subscription_id) = subscription_id mod 5 because this hash function is simple to implement and has a uniform distribution of hash values, which means that the probability of a collision is relatively low. However, the second hash function is more complex to implement and doesn't have a uniform distribution of hash values, so it is not efficient for space and also access time to data is more than the first one.

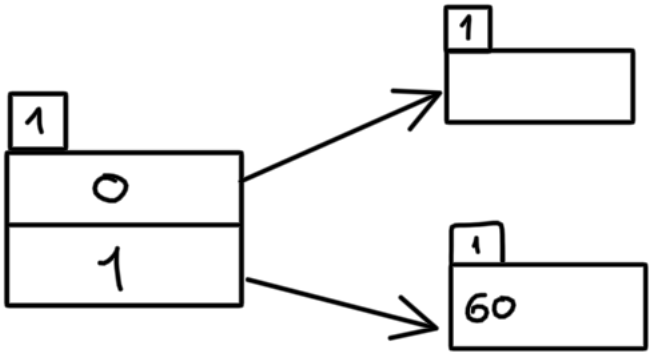
Question 2

Initially global-depth and local-depth are 1. The small box on the big box represents the depth. The frame looks like:



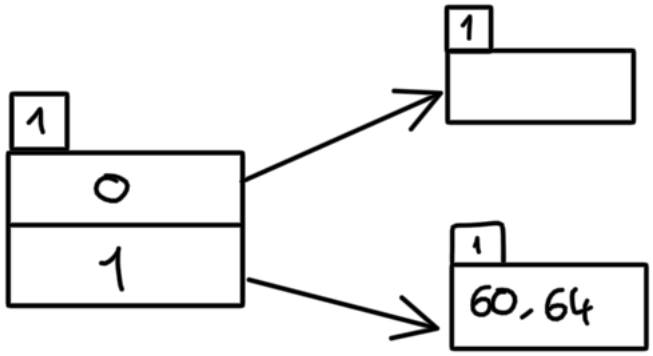
Insert 60:

60 mod 29 = 2, and 2 is represented 10 in binary. The leftmost digit is 1, so it should be pointed by 1. The frame will look like:



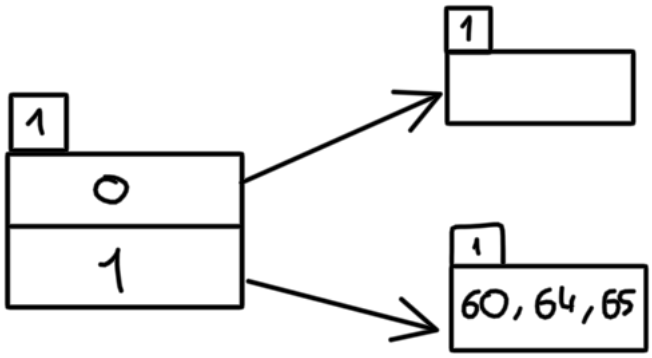
Insert 64:

64 mod 29 = 6, and 6 is represented 110 in binary. The leftmost digit is 1, so it should be pointed by 1. The frame will look like:



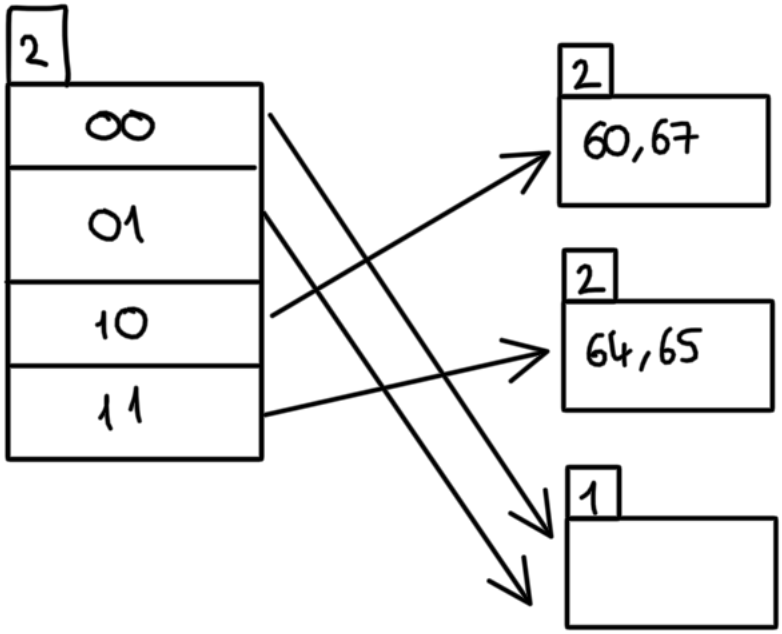
Insert 65:

65 mod 29 = 7, and 7 is represented 111 in binary. The leftmost digit is 1, so it should be pointed by 1. The frame will look like:



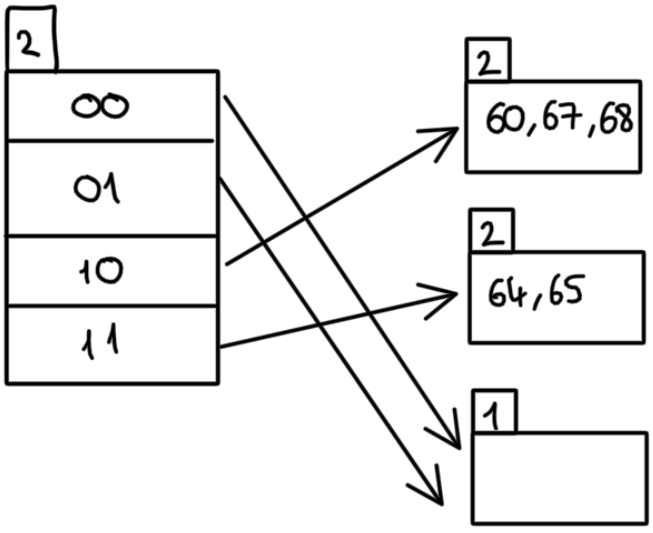
Insert 67:

67 mod 29 = 9, and 9 is represented 1001 in binary. The leftmost digit is 1, so it should be pointed by 1. However, the capacity of bucket is 3, so there is an overflow. We should split the bucket and expand the directory because local-depth equals to global-depth. Also, we need to rehash the numbers in overflowing bucket after the split. Global-depth is increased by 1. Other bucket will stay untouched, it is the bucket for records which starts with 0. For rehashing, 60, 64, 65 and 67 will be pointed by 10, 11, 11, 10 respectively because of their 2 leftmost digits. The frame will look like:



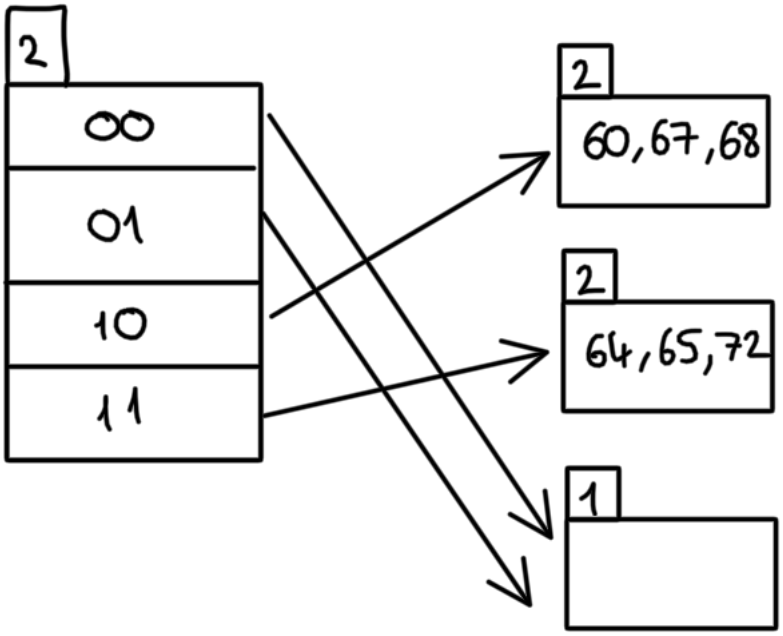
Insert 68:

68 mod 29 = 10, and 10 is represented 1010 in binary. The 2 leftmost digits are 10, so it should be pointed by 10. The frame will look like:



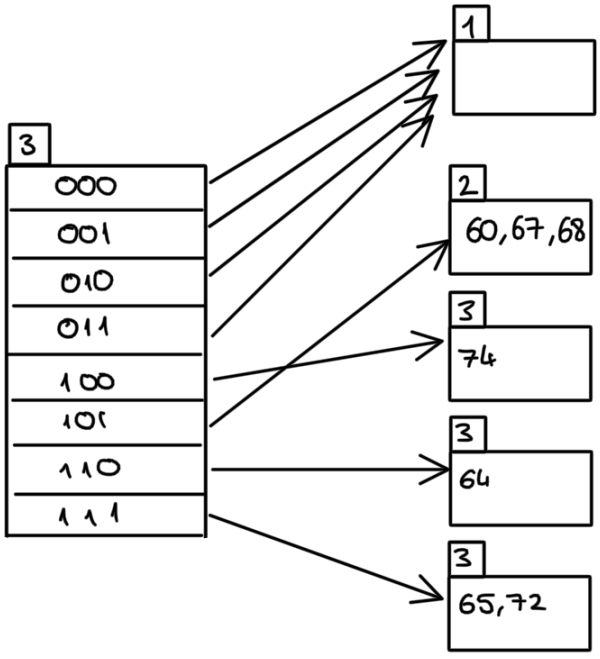
Insert 72:

72 mod 29 = 14, and 14 is represented 1110 in binary. The 2 leftmost digits are 11, so it should be pointed by 11. The frame will look like:



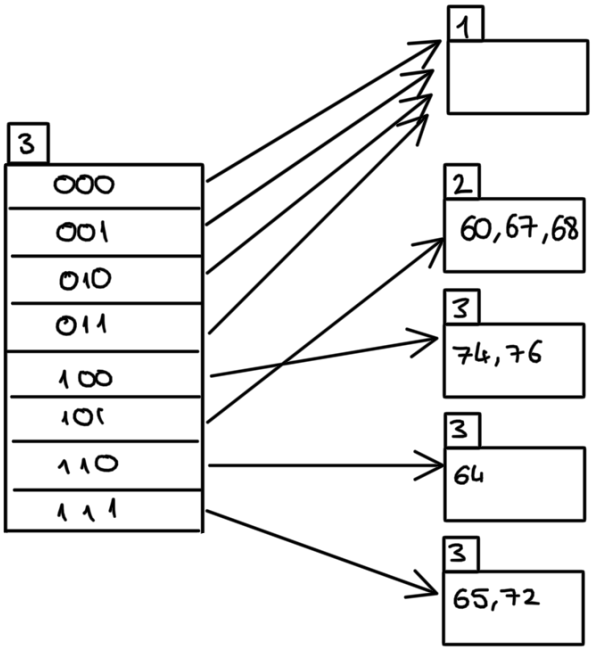
Insert 74:

74 mod 29 = 16, and 16 is represented 10000 in binary. The 2 leftmost digits are 10, so it should be pointed by 10. However, the capacity of bucket is 3, so there is an overflow. We should split the bucket and expand the directory because local-depth equals to global-depth. Also, we need to rehash the numbers in overflowing bucket after the split. Global-depth is increased by 1. For rehashing, 64, 65, 72 and 74 will be pointed by 110, 111, 111, 100 respectively because of their 3 leftmost digits. The frame will look like:



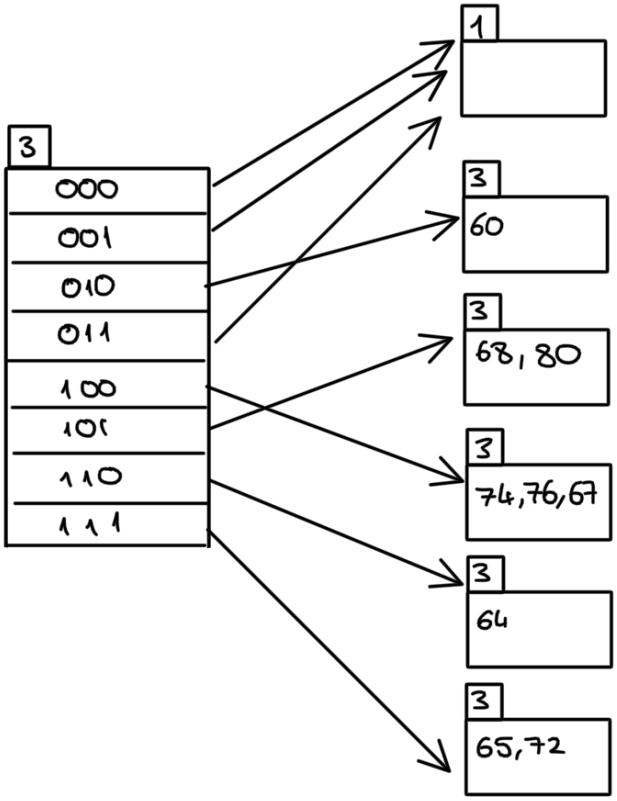
Insert 76:

76 mod 29 = 18, and 18 is represented 10010 in binary. The 3 leftmost digits are 100, so it should be pointed by 100. The frame will look like:



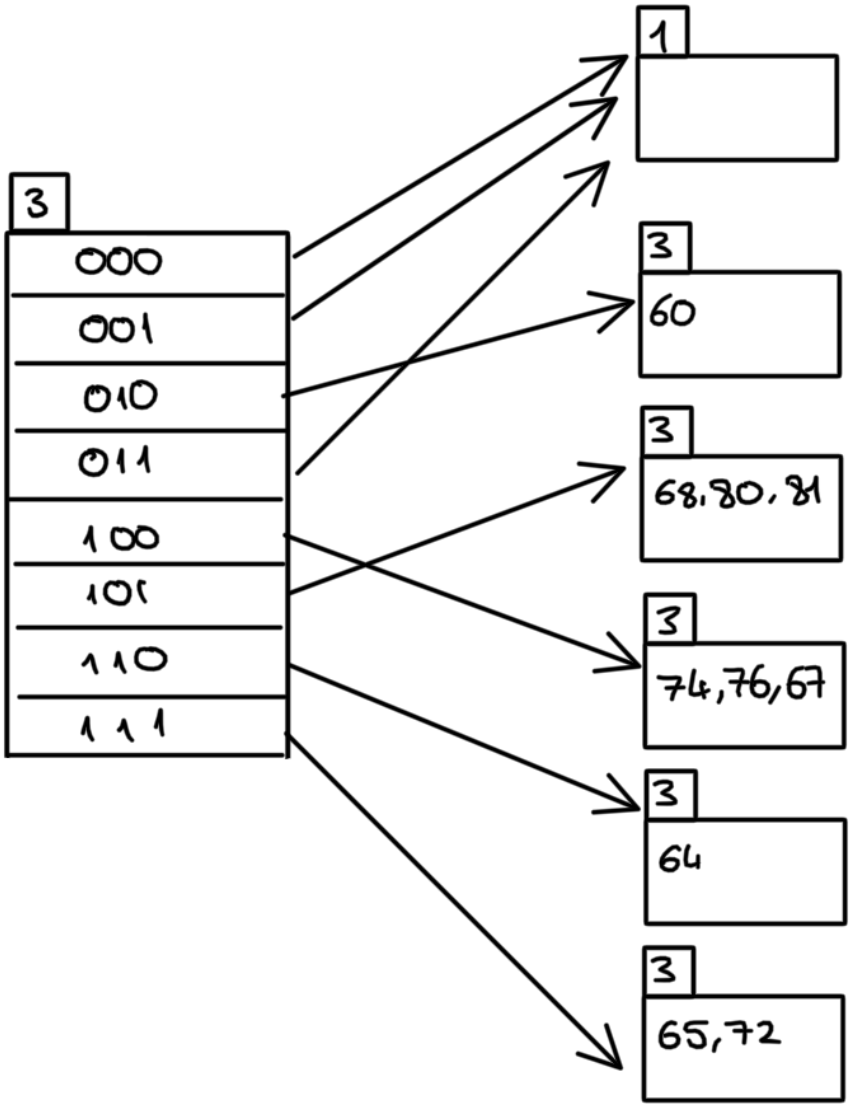
Insert 80:

80 mod 29 = 22, and it is represented 10110 in binary. The 3 leftmost digits are 101, so it should be pointed by 101. However, there is an overflow. We only need to split the bucket because local-depth is less than the global-depth (2 < 3). Therefore, we do not need to expand the directory. After rehashing, 60, 67, 68, 80 will be pointed by 010, 100, 101, 101 respectively because of their 3 leftmost digits. The frame will look like:



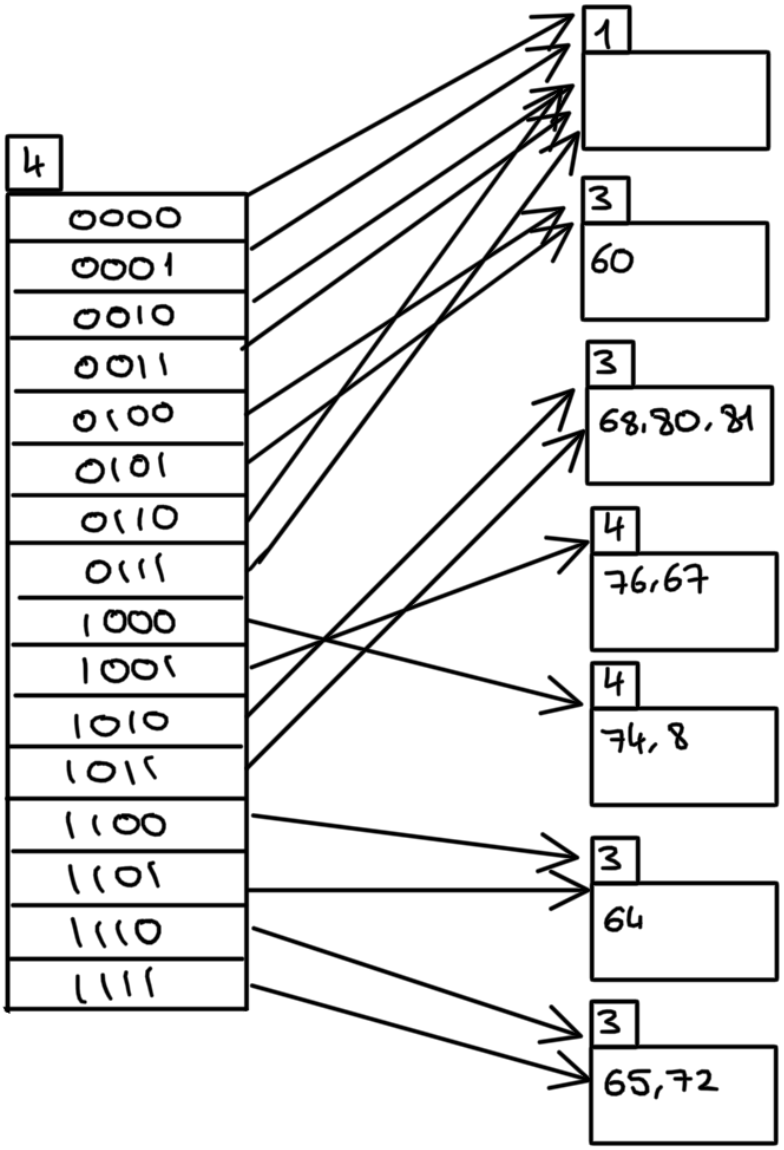
Insert 81:

81 mod 29 = 23, and it is represented 10111 in binary. The 3 leftmost digits are 101, so it should be pointed by 101. The frame will look like:



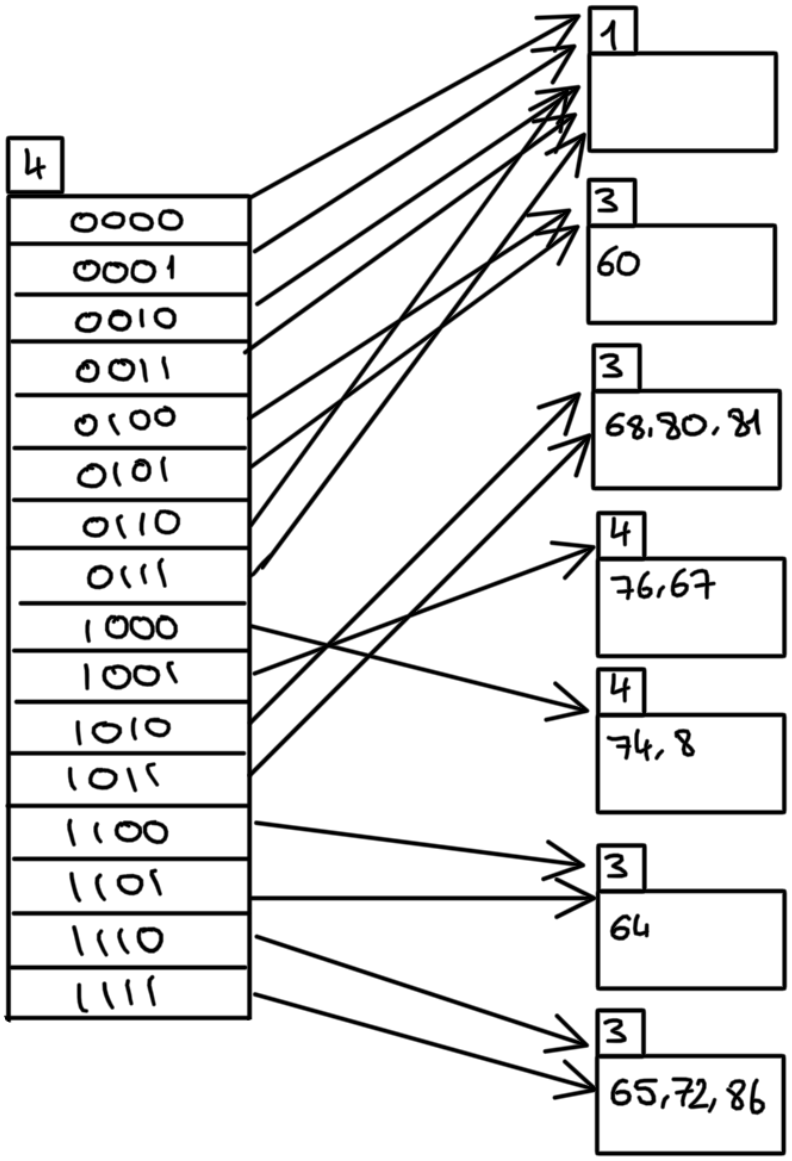
Insert 8:

8 mod 29 = 8, and it is represented 1000 in binary. The 3 leftmost digits are 100, so it should be pointed by 100. However, it causes overflow in the bucket. Because the local-depth is equal to the global-depth, directory expansion takes place along with bucket splitting. Also, the elements is overflowing bucket which are 74, 76, 67, 8 are rehashed, and they will be pointed by 1000, 1001, 1001, 1000 respectively because of their 4 leftmost digits. The frame will look like:



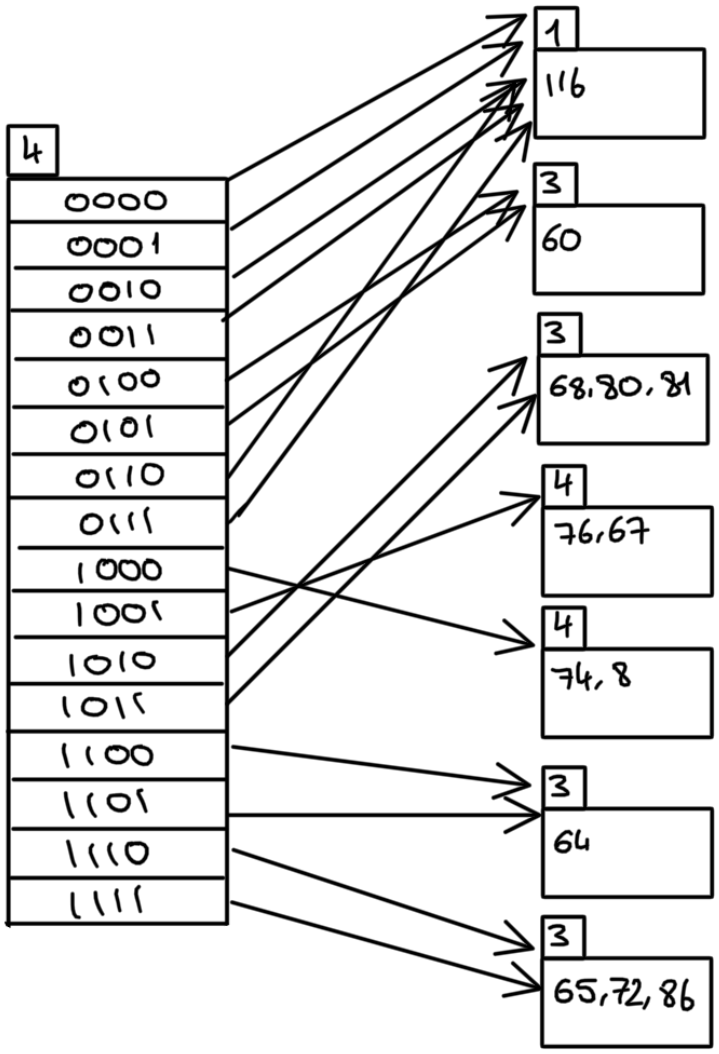
Insert 86:

86 mod 29 = 28, and it is represented as 11100 in binary. The 4 leftmost digits are 1110, so it should be pointed by 1110. The frame will look like:



Insert 116:

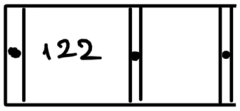
116 mod 29 = 0, and it is represented as 0000 in binary. It is pointed by 0000. The frame will look like:



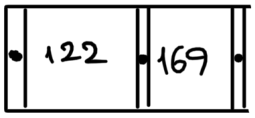
Insertion of the values is completed.

Question 3

Insert 122:

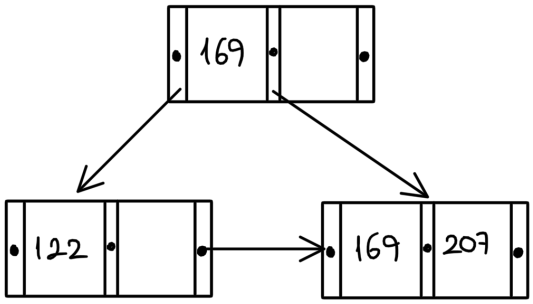


Insert 169:



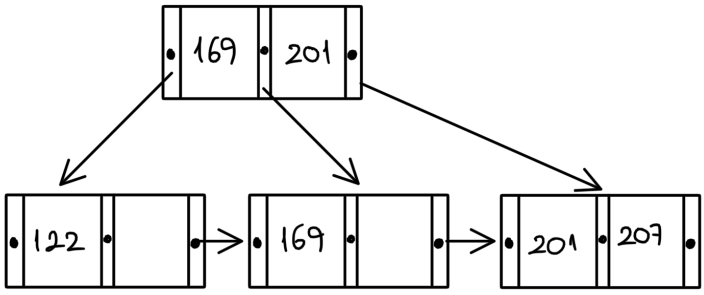
Insert 207:

The node is full, so we must split the node by moving 169 to a new node. In addition, we create a parent node consisting of the mid key 169.



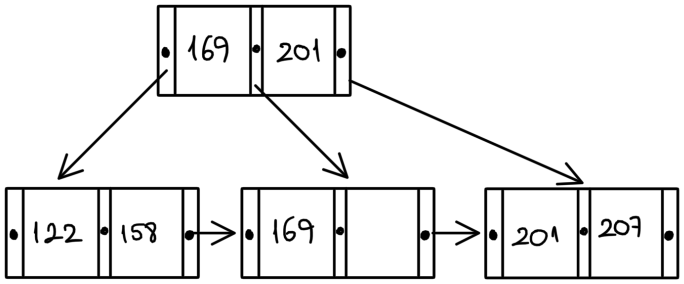
Insert 201:

It should be located to the left of 207, but again the node is full. We split the node by moving 207 to a new node. We also move mid key 201 to the parent node.



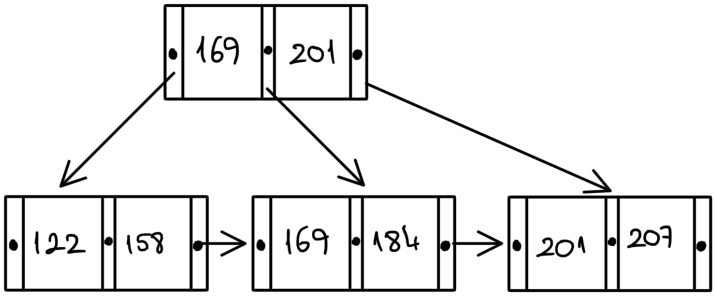
Insert 158:

122 < 169, we can insert it to right of 122.



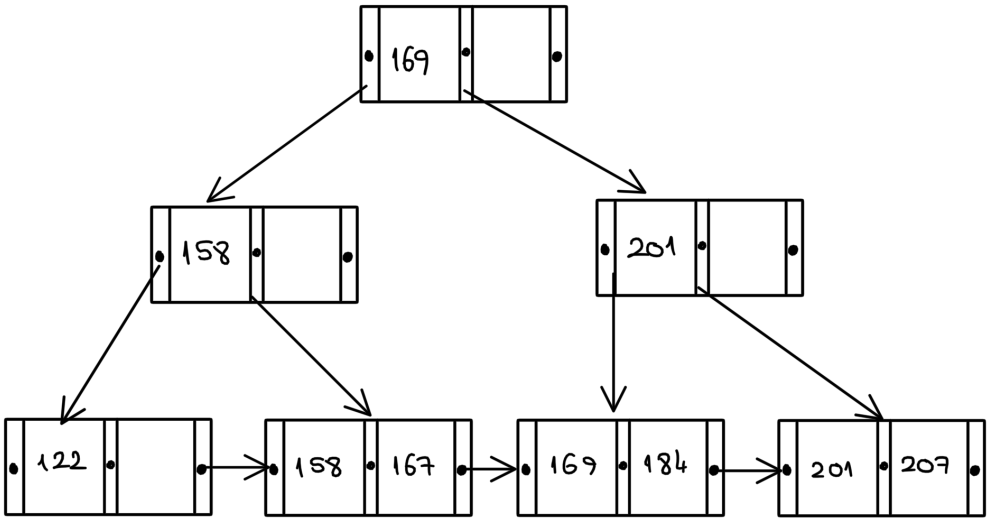
Insert 184:

It is between 169 and 201, so it should be placed the node pointed by mid, right of 169.



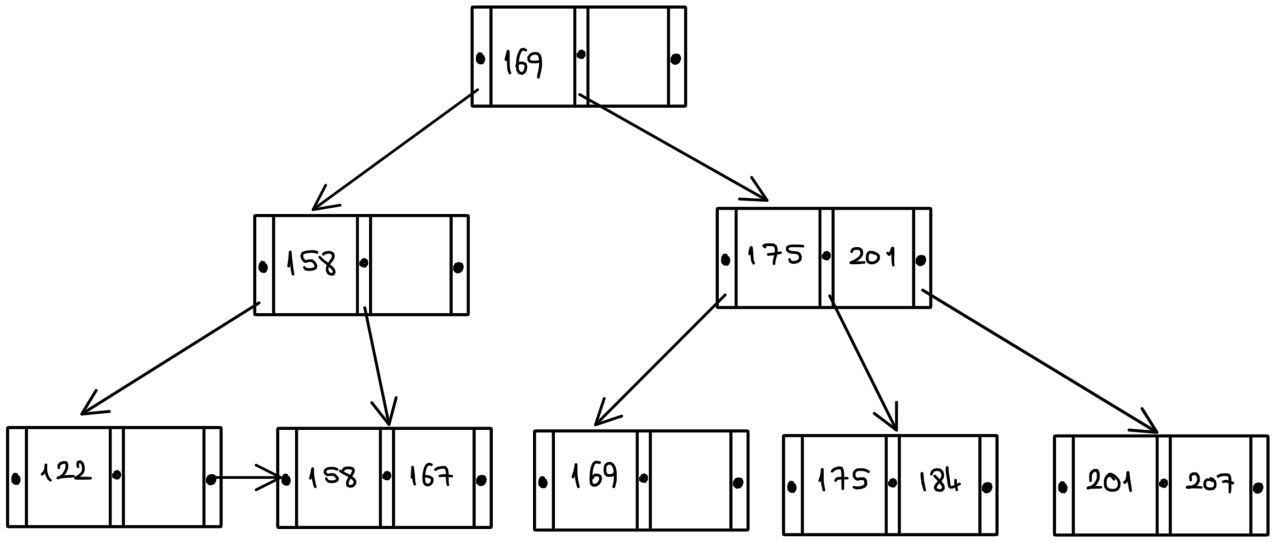
Insert 167:

It should be placed right of 158, but the node is full. Therefore, we need to split it by moving 158 to a new node. The middle key is 158, so copy it up and move it to the parent node. However, the parent is also full, so split it and copy up the middle key, and create a parent node with it.



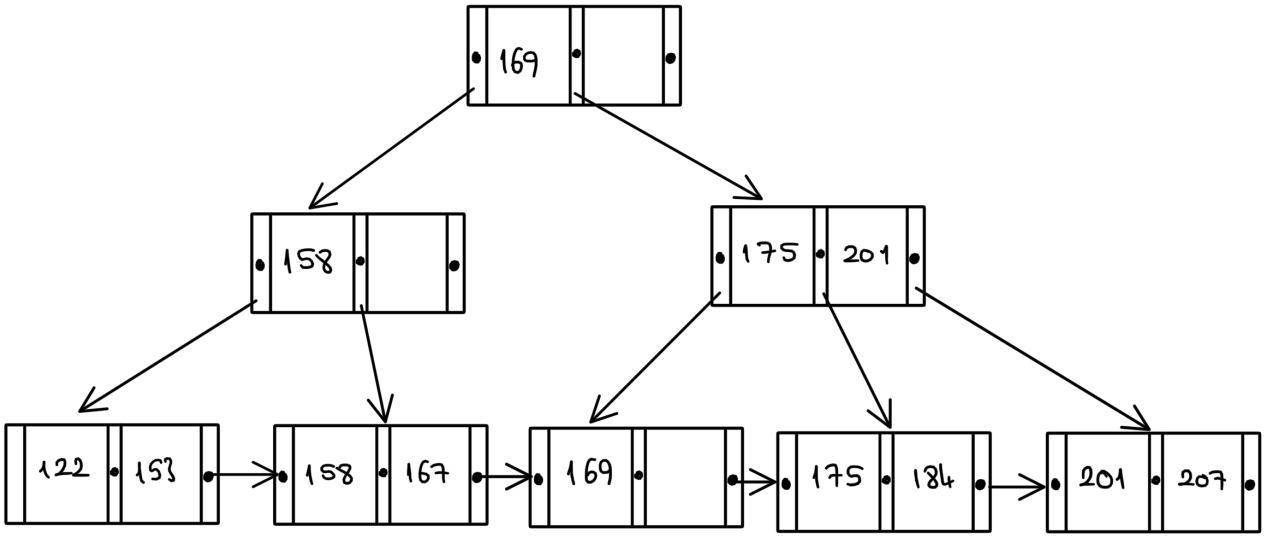
Insert 175:

It should be placed right of 169; however, the node is full. Thereby, we need to split it by moving 184 to a new node, and because 175 is mid key, move it to the parent.



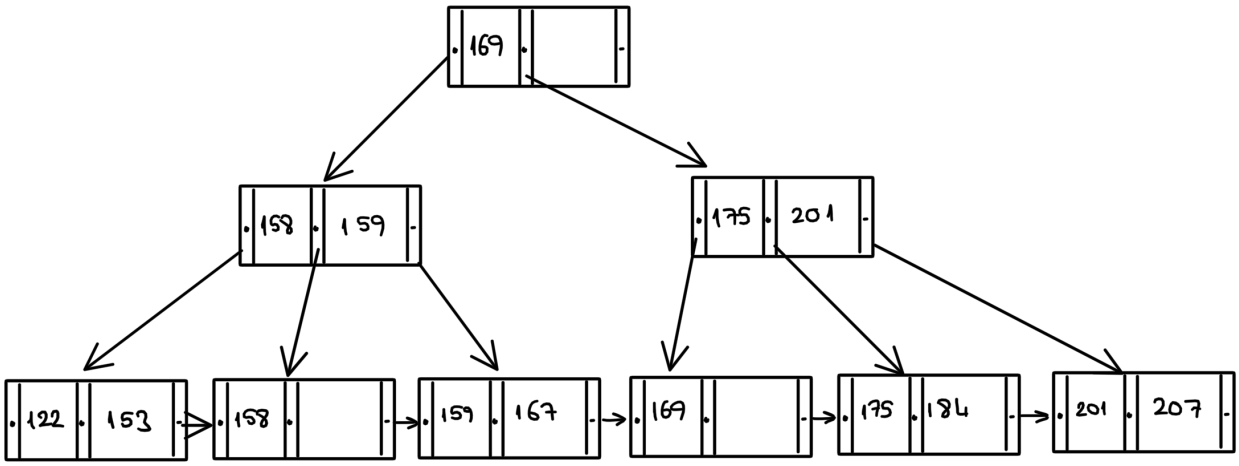
Insert 153:

153 < 169, and it is less than 158, so it should be placed right of 122.



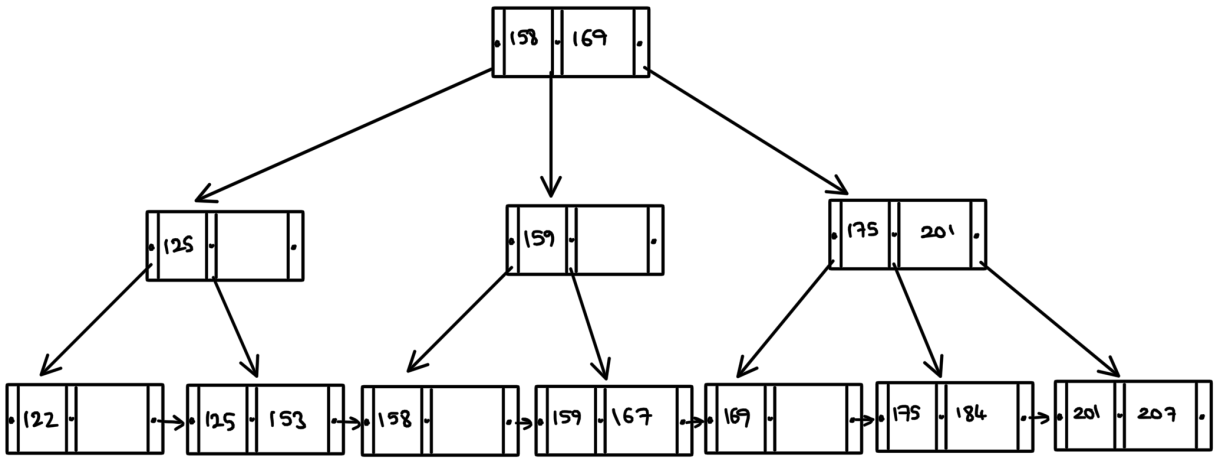
Insert 159:

It should be placed right of 158, but the node is full. Therefore, we need to split it by moving 167 to a new node. Because the middle key is 159, we move it to the parent node.



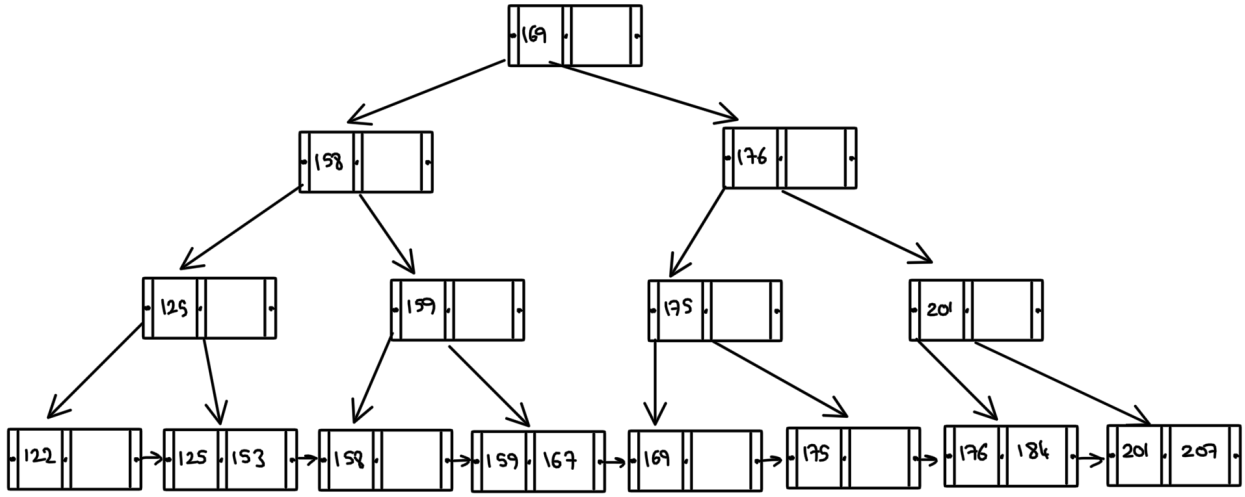
Insert 125:

It should be placed at the left of 153, but the node is full. Thereby, we need to split it by moving 153 to a new node, and also move the mid key 125 to the parent node. However, the parent node is also full, so we need to split it and move the mid key 158 to the parent node. Then we need to redistribute the keys because its number of children should be equal to number of keys + 1.



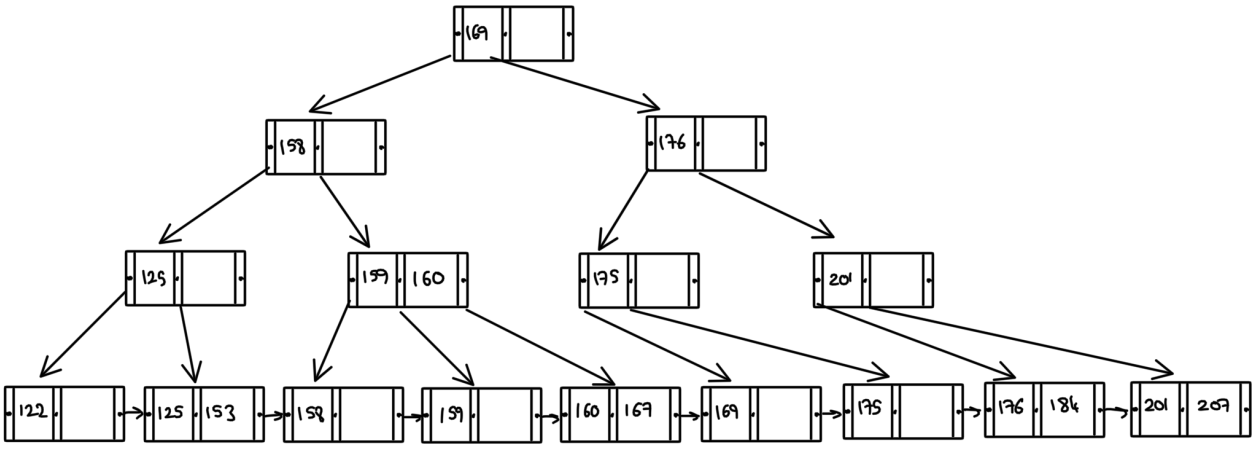
Insert 176:

It should be placed right of 175, but the node is full. Therefore, we need to split it by moving 184 to a new node. Because 176 is mid key, it should be moved to the parent node; however, it is full, so we need to split it and mid key 176 should be moved to the parent node. However, it is full again, and it is the root. Thereby, we need to split it and create a new parent with mid key 169. Then redistribute the key values.



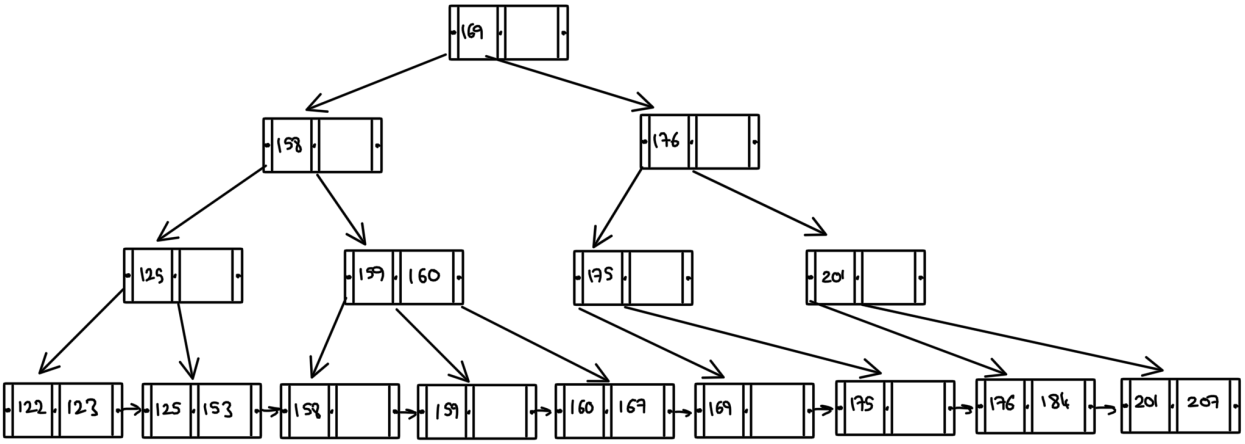
Insert 160:

It should be placed right of 159, but the node is full. Therefore, we need to split it by moving 167 to a new node, also move the mid key 160 to the parent node.

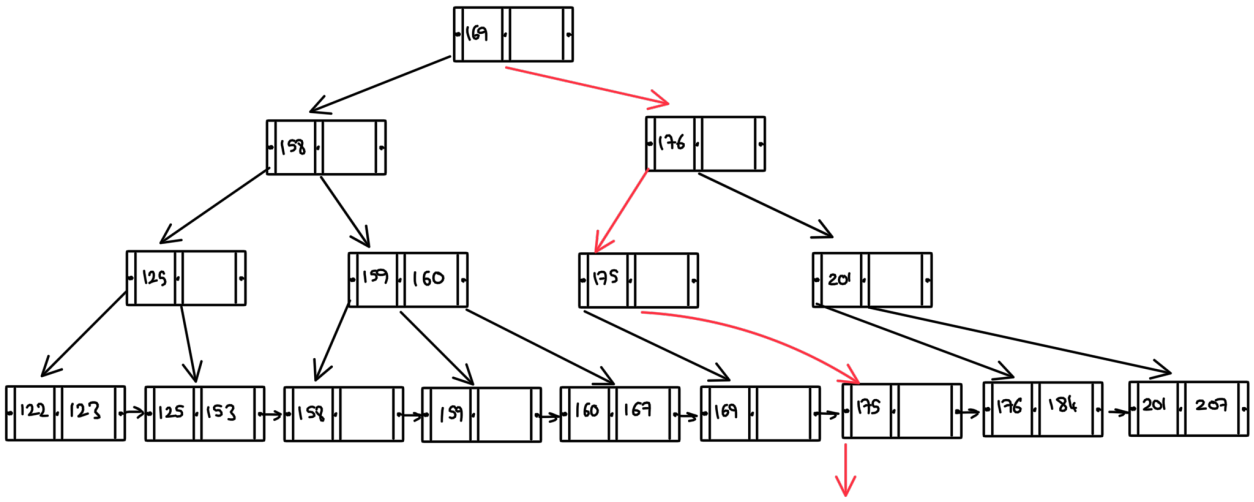


Insert 123:

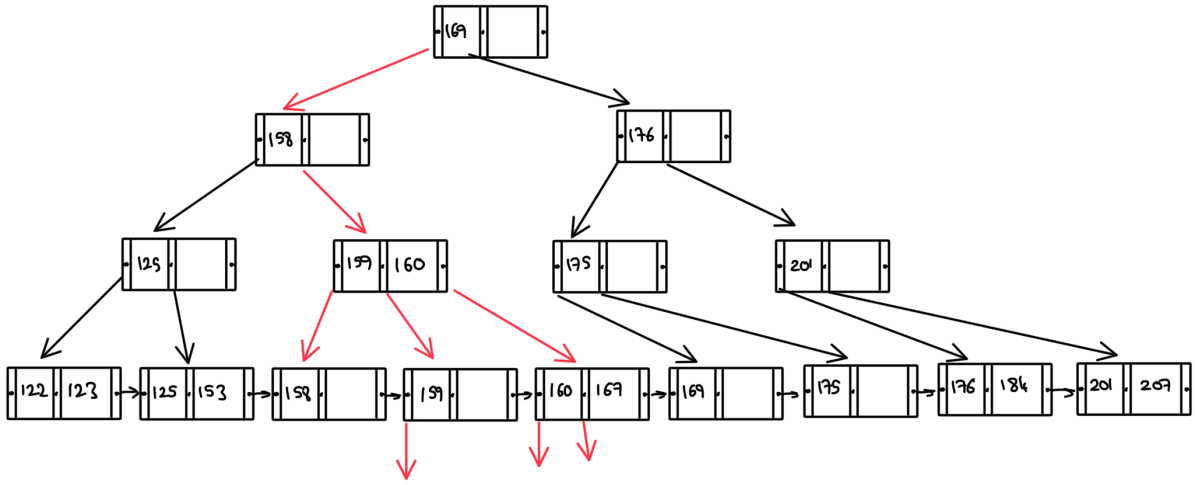
123 < 169 & 123 < 158 & 123 < 125, so it should be placed at right of 122.



- **Find records with a search-key value of 175.**



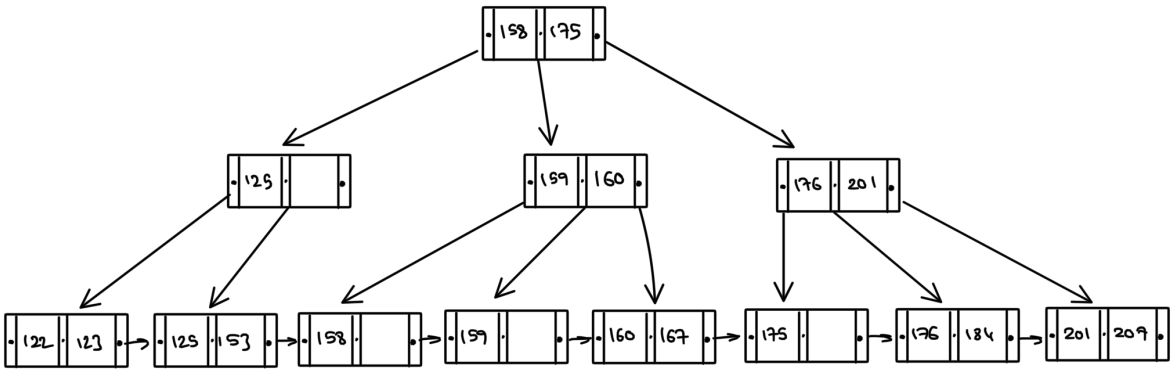
- **Find records with a search-key value between 159 and 167, inclusive.**



- **Show the form of the tree after deleting key values 169, 175, 122 and 160.**

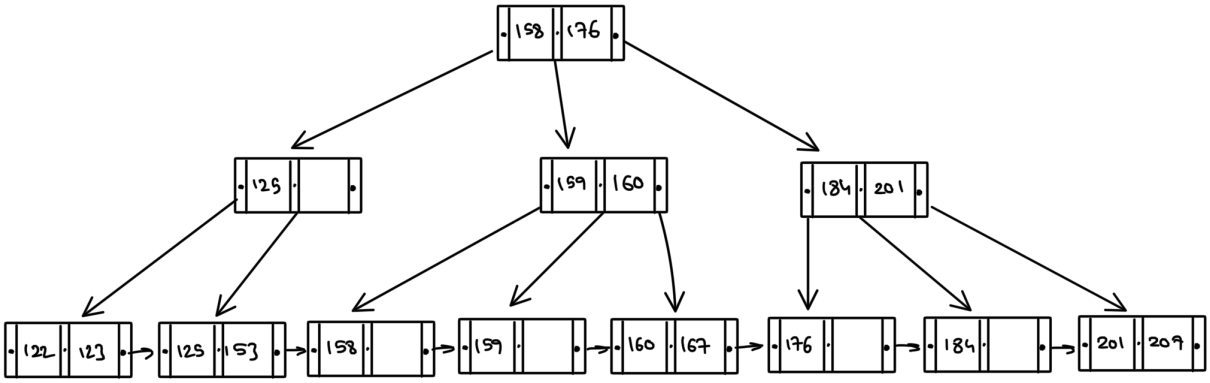
Delete 169:

169 is founded, then removed from the leaf. Then, 176 is merged with 201 and 175 is merged with the root 158 because 169 is also root.



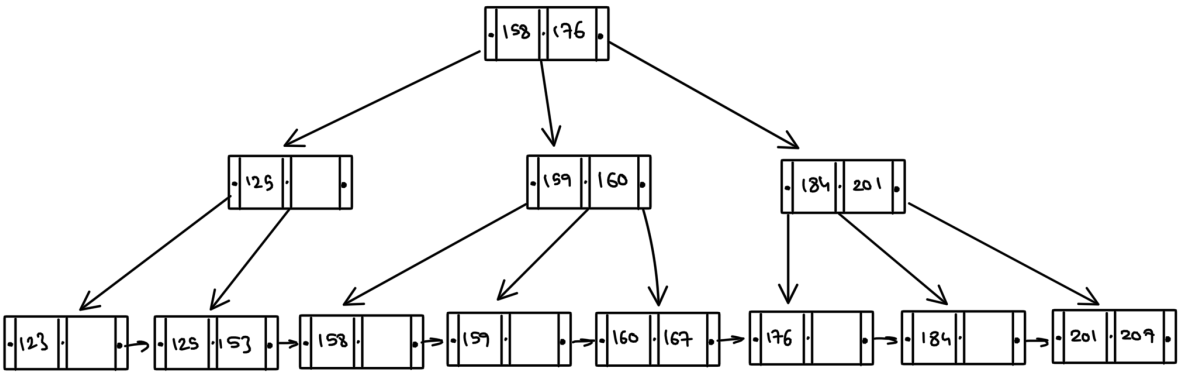
Delete 175:

After removing 175 from the leaf, 176 is borrowed from the sibling and 184 is moved to the parent node. Also, 176 is moved to the root.



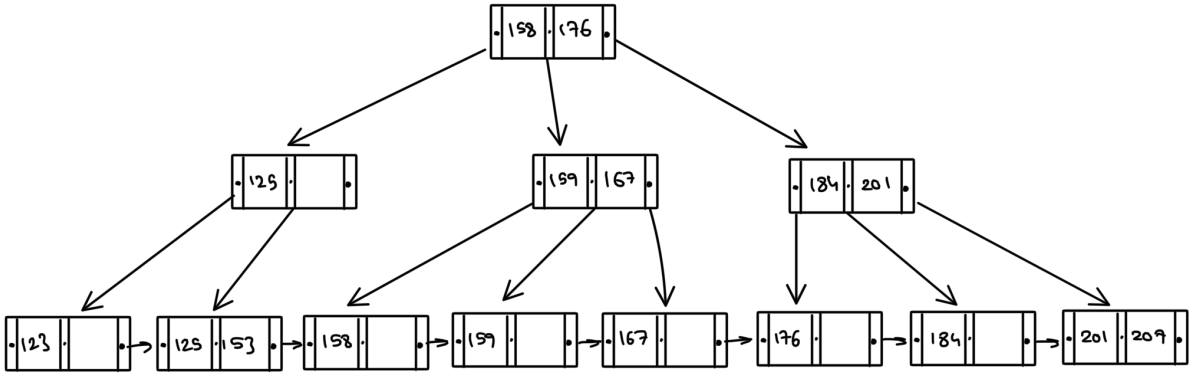
Delete 122:

Simply delete the leaf because 122 is not an internal node.



Delete 160:

Delete 160 from the leaf, then move the 167 to the parent.



Question 4

1. **Show the structure of the database when the given table is indexed using a clustering index with group ID.**

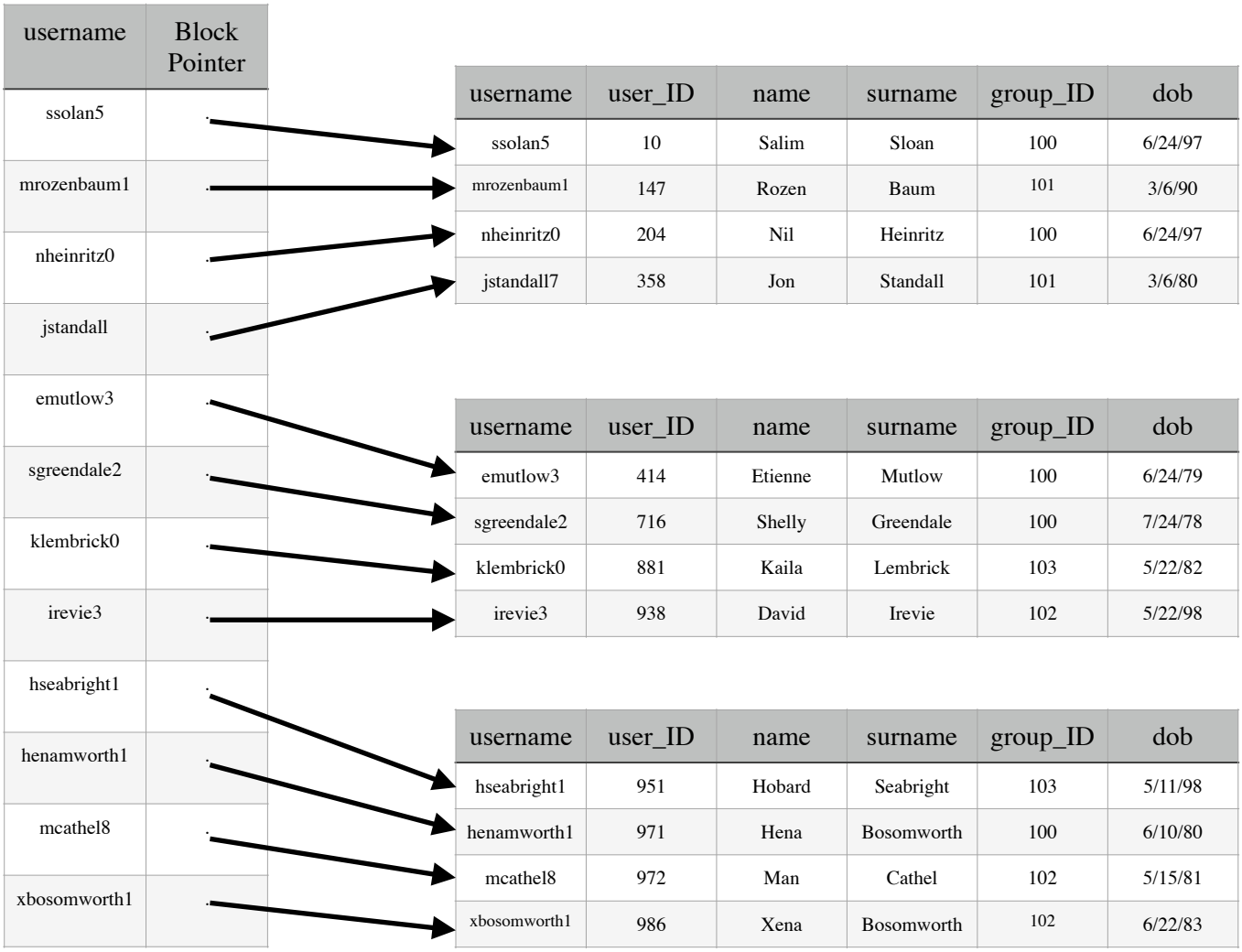
group_ID	Block Pointer
100	
101	
102	
103	

group_ID	user_ID	name	surname	username	dob
100	10	Salim	Sloan	ssolan5	6/24/97
100	204	Nil	Heinritz	nheinritz0	6/24/97
100	414	Etienne	Mutlow	emutlow3	6/24/79
100	716	Shelly	Greendale	sgreendale2	7/24/78

group_ID	user_ID	name	surname	username	dob
100	971	Hena	Bosomworth	henamworth1	6/10/80
101	147	Rozen	Baum	mrozenbaum1	3/6/90
101	358	Jon	Standall	jstandall7	3/6/80
102	938	David	Irevie	irevie3	5/22/98

group_ID	user_ID	name	surname	username	dob
102	972	Man	Cathel	mcathel8	5/15/81
102	986	Xena	Bosomworth	xbosomworth1	6/22/83
103	881	Kaila	Lembrick	klembrick0	5/22/82
103	951	Hobard	Seabright	hseabright1	5/11/98

2. Show the structure of the database when the given table is indexed using a secondary key username.



username	Block Pointer
ssolan5	.
mrozenbaum1	.
nheinritz0	.
jstandall	.
emutlow3	.
sgreendale2	.
klembrick0	.
irevie3	.
hseabright1	.
henamworth1	.
mcathel8	.
xbosomworth1	.

username	user_ID	name	surname	group_ID	dob
ssolan5	10	Salim	Sloan	100	6/24/97
mrozenbaum1	147	Rozen	Baum	101	3/6/90
nheinritz0	204	Nil	Heinritz	100	6/24/97
jstandall7	358	Jon	Standall	101	3/6/80

username	user_ID	name	surname	group_ID	dob
emutlow3	414	Etienne	Mutlow	100	6/24/79
sgreendale2	716	Shelly	Greendale	100	7/24/78
klembrick0	881	Kaila	Lembrick	103	5/22/82
irevie3	938	David	Irevie	102	5/22/98

username	user_ID	name	surname	group_ID	dob
hseabright1	951	Hobard	Seabright	103	5/11/98
henamworth1	971	Hena	Bosomworth	100	6/10/80
mcathel8	972	Man	Cathel	102	5/15/81
xbosomworth1	986	Xena	Bosomworth	102	6/22/83

In general, these arrows are like creeper. However, to ease to read we arranged to order of usernames.