National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Advance Database Concepts**	**Course Code:**	**CS4064**
	Program:	**BS (Computer Science)**	**Semester:**	**Spring 2023**
	Out Date:	**8-Mar-2023**	**Total Marks:**	
	Due Date:	**Thu 16-Mar-2023** ***(Start of class)***	**Weight:**	
	Section		**Page(s):**	**2**
	Assignment:	**2 (File Structures & Hashing) Solution**		

Instructions:

- *This assignment is an individual assignment.*
- *You are required to submit the hard copy of your assignment at the start of your class.*
- *Use any valid assumption where needed.*
- *For any query please contact your TA.*

Q1. Consider a file system on a disk with block size *B=1024 bytes*. A file has *r=10,000,000* STUDENT records of fixed length (un-spanned). Each record length is *100 bytes*. Assume there are 200 departments and 50,000 students per department. Primary key of student file is RollNo.
Estimate the number of block fetches needed to compute the following queries:

a. SELECT * FROM STUDENT WHERE rollno=1357; (Assume file is not ordered)
b. SELECT * FROM STUDENT WHERE rollno=2468; (Assume file is ordered on rollno)
c. SELECT * FROM STUDENT WHERE deptno= 11; (Assume file is not ordered)
d. SELECT * FROM STUDENT WHERE deptno= 22; (Assume file is ordered on deptno)

Ans:

R = 100 bytes
bfr = floor(B/R) = floor(1024/100) = 10
number of blocks required = ceil(r/bfr) = ceil(10,000,000/10) = 1,000,000 blocks
a) b = 1,000,000
b) $\log_2(b) = \log_2(1{,}000{,}000) = 20$
c) b = 1,000,000
d) b1 = r1/bfr = 50,000/10 = 5000; $\log_2(b)=20$; total block fetches = 20 + 5000 - 1 = 5019 blocks

Q2. A customer file has following customer ID values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

a. Consider these customer IDs as hash key values. The file uses 6 buckets named 0 to 5. One bucket cannot hold more than 3 records, means at max a bucket can hold 3 records. Load these records in file using hash function h(k) = k mod 6, in the given order.
b. Calculate the average number of block accesses for random retrieval on customer ID.
c. Load the given values in expandable hash files based on extendible hashing, show structure on each step, use hash function h(k) = K mod 7, max 3 records can be kept in one bucket.
d. Load the given values in expandable hash files based on dynamic hashing, show structure on each step, use hash function h(k) = K mod 7, max 3 records can be kept in one bucket.

e. Load the given values in expandable hash files based on linear hashing, show structure on each step, use hash function h(k) = K mod 7, max 3 records can be kept in one bucket.

f. Show the structure after inserting an entry with customer ID value 13 into the final structure of part (c).

g. Show the structure after inserting an entry with customer ID value 18 into the structure of part (f).

Answer:

a.

HashInd	Buckets
0	
1	7, 19, 31
2	2
3	3
4	
5	5, 11, 17, **23,29 (Overflow occured)**

→ Overflow occurred at HashIndVal = 5, since two hash entries collided due to limited buckets.

b.

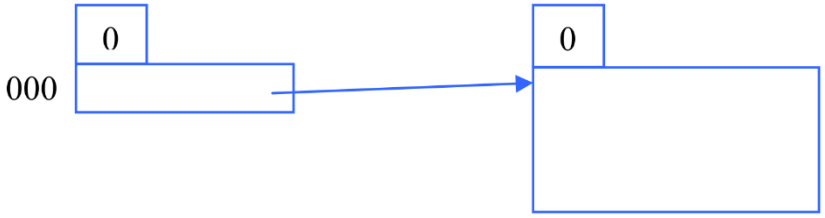
ID.

There are two values that overflowed the bucket 6. I.e 23 and 29. And 8 values placed without any collision or overflow. Hence,

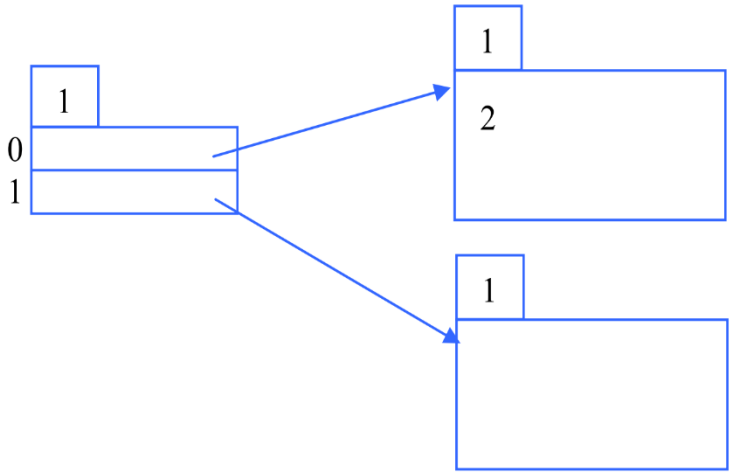
= 1 (8/10) + 2 (2/10)

Average Block Access = (1 * 0.8) + (2 * 0.2)

c. The initial hash structure looks like this:



When the record 2 is added, the hash structure is extended to look like this:



Note that 2 mod 7 = 2 = 010 – the first bit is 0, so it goes in the bucket pointed to by the 0 slot of the bucket address table. Likewise, the following records can be added to the hash structure as it stands:

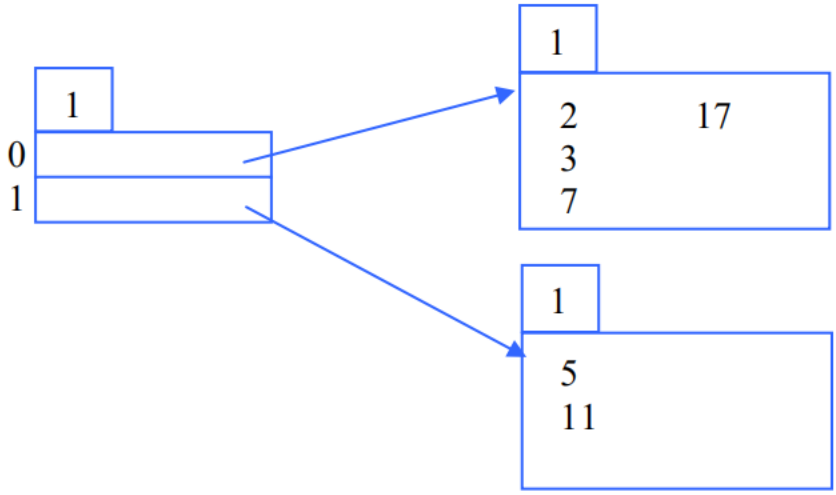
Add 3 → 3 mod 7 = 3 = 011, goes to bucket 0

Add 5 → 5 mod 7 = 5 = 101, goes to bucket 1

Add 7 → 7 mod 7 = 0 = 000, goes to bucket 0

Add 11 → 11 mod 7 = 4 = 100, goes to bucket 1

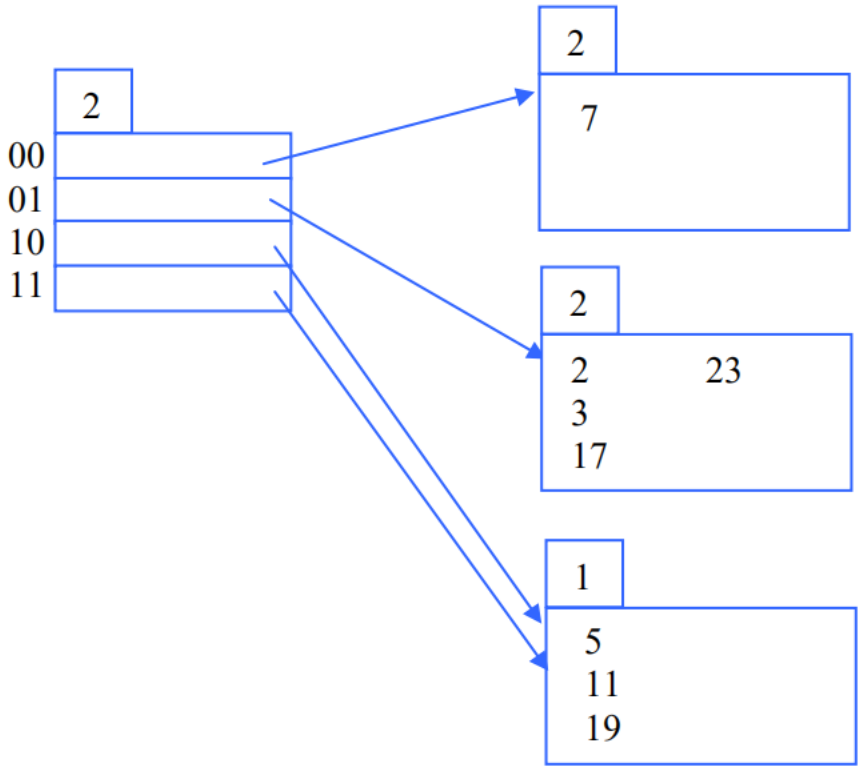
Add 17 → 17 mod 7 = 3 = 011, goes to bucket 0



At this point, bucket 0 has overflowed, so we must split the bucket and rehash its contents using 2 bits. After doing that, we add the following records:

Add 19 → 19 mod 7 = 5 = 101, goes to bucket 10

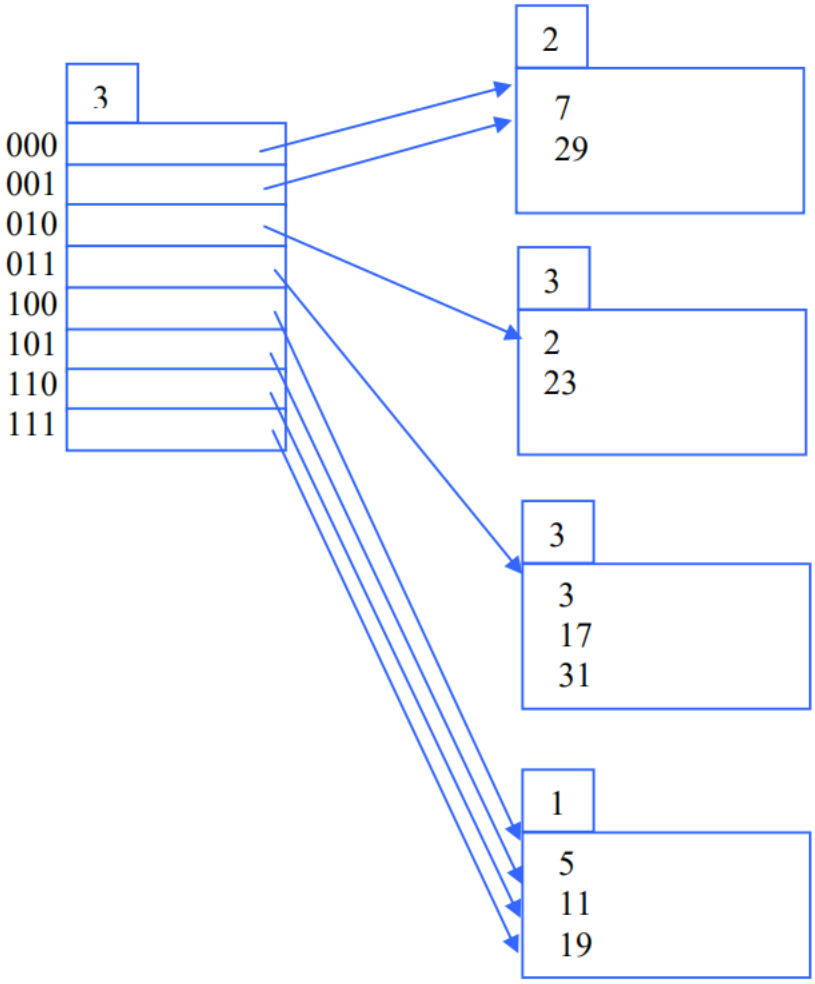
Add 23 → 23 mod 7 = 2 = 010, goes to bucket 01



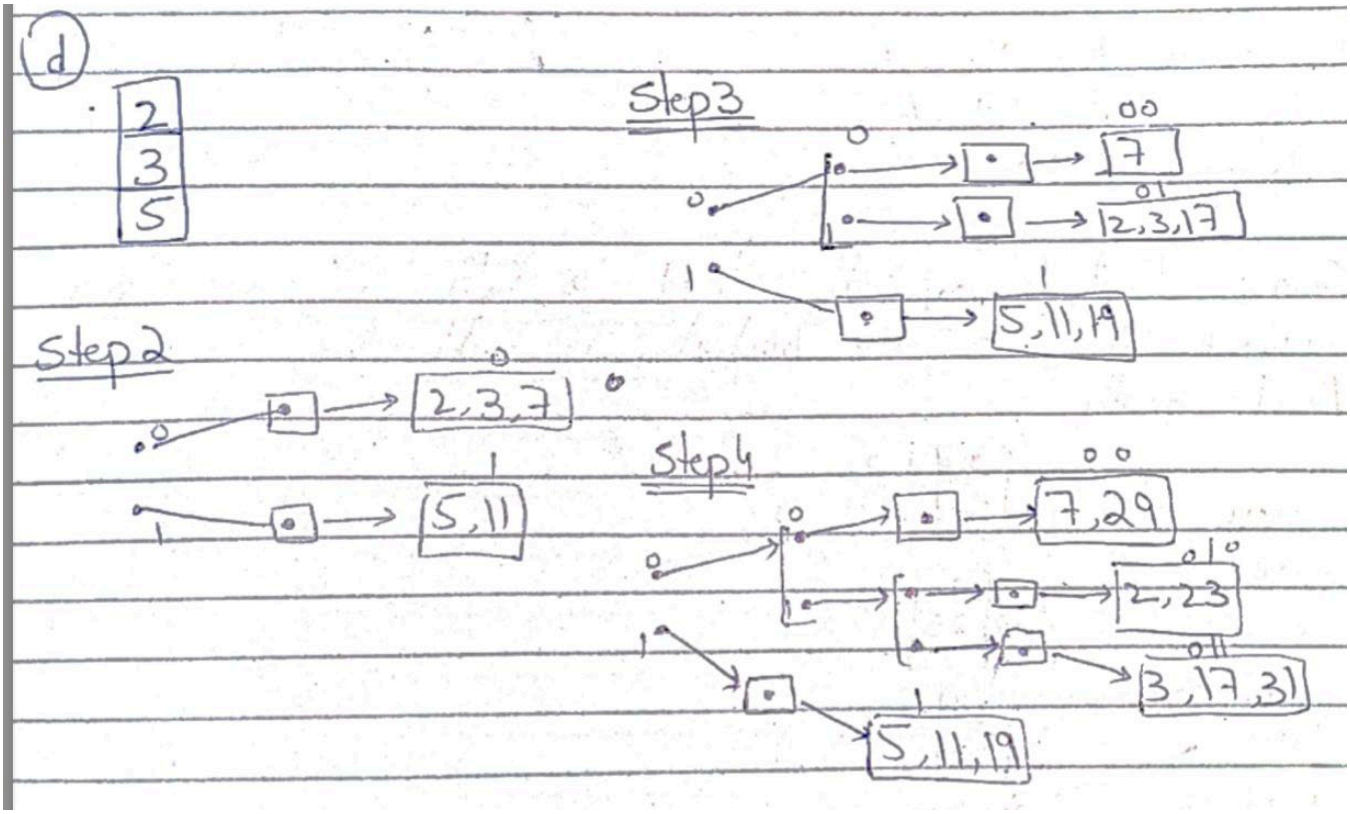
At this point, bucket 01 has overflowed, so we must split the bucket and rehash its contents using 3 bits. This allows us to add the remaining records, giving us the following hash structure:

Add 29 → 29 mod 7 = 1 = 001, goes to bucket 001

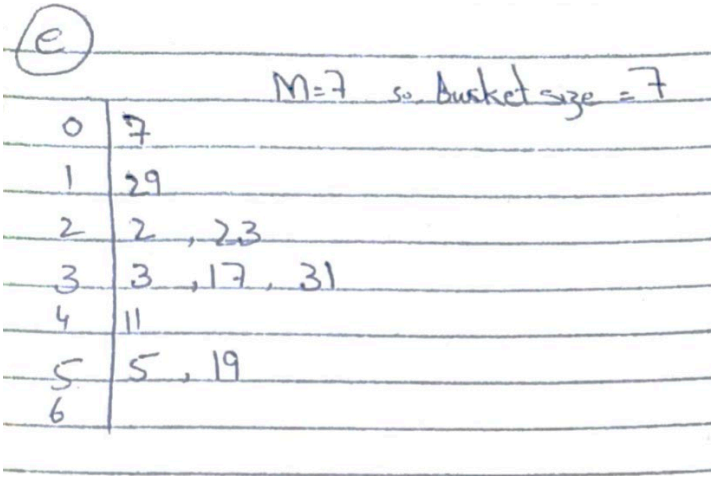
Add 31 → 31 mod 7 = 3 = 011, goes to bucket 011



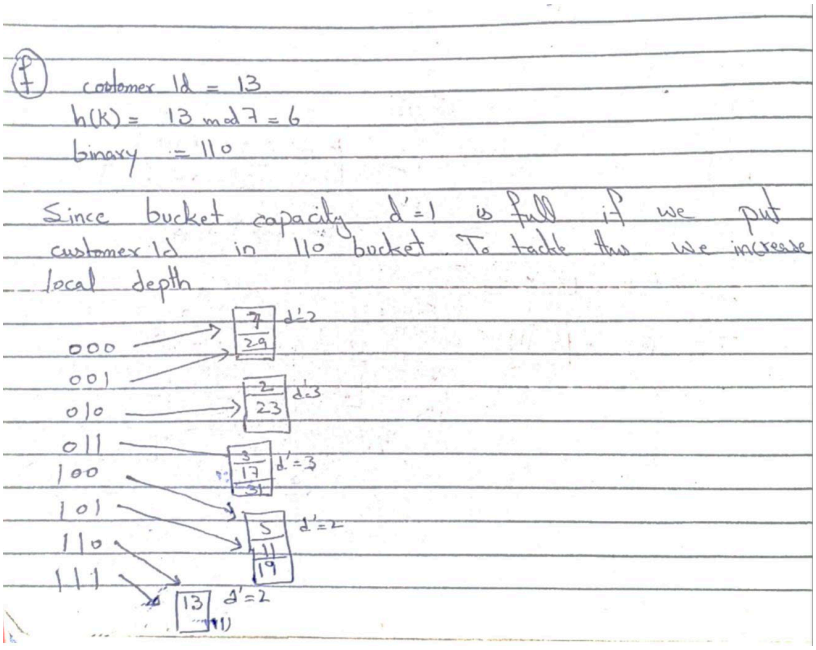
d.


d
2
3
5
Step3
00
7
01
2,3,17
1
5,11,19
Step2
2,3,7
1
5,11
Step4
00
7,29
010
2,23
011
3,17,31
1
5,11,19

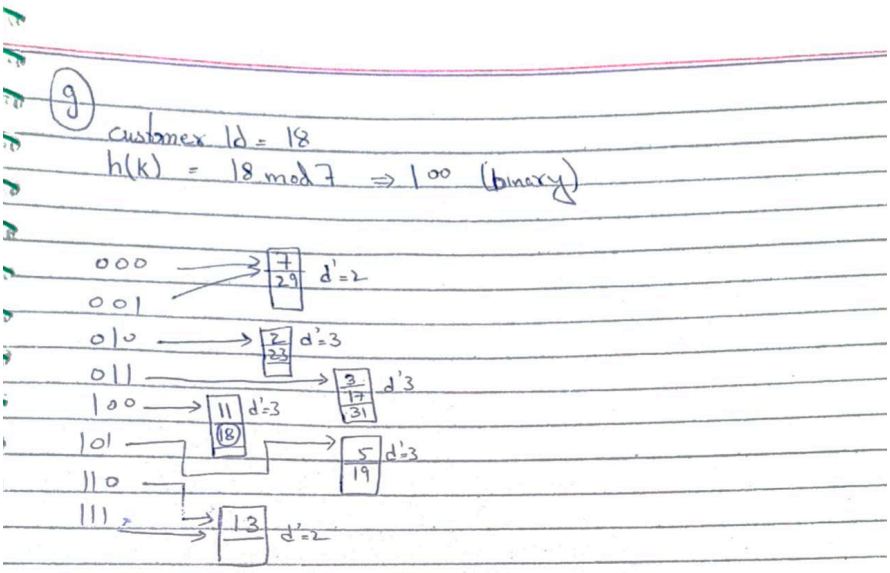
e.


e
M=7 so Bucket size = 7
0 | 7
1 | 29
2 | 2, 23
3 | 3, 17, 31
4 | 11
5 | 5, 19
6 |

f.



g.



Q3. Consider a dynamic hash structure where buckets can hold up to three records. Initially the structure is empty. Then we insert the following records, in the order below, where we indicate the hashed key in brackets (in binary):

a	[010000]
b	[011010]
c	[111100]
d	[001110]
e	[010111]

f	[011010]
g	[101001]
h	[010111]
i	[000110]
j	[101001]

Show the linear hash structure after these records have been inserted. Assume that the threshold value is 2. (i.e., when the average number of keys per non-overflow bucket is greater than 2, we allocate another bucket).

Ans: