1. Which collision resolution technique involves using linked lists to store multiple values in a single hash table bucket?

**A) Separate Chaining**

B) Linear Probing

C) Quadratic Probing

D) Double Hashing

2.

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> hashTable(10, -1);

hashTable[3] = 42;

hashTable[6] = 19;

hashTable[9] = 56;

cout << hashTable[5] << " " << hashTable[6] << endl;

return 0;

}

What will be the output of the program?

**A) -1 19**

B) 42 19

C) -1 -1

D) 19 56

3. In which collision resolution technique is the next available slot (bucket) searched in a linear manner when a collision occurs?

A) Separate Chaining

**B) Linear Probing**

C) Quadratic Probing

D) Double Hashing

4. For the given hash table, in what location will the element 58 be hashed using quadratic probing?

0 | 49

1 |

2 |

3 |

4 |

5 |

6 |

7 |

8 | 18

9 | 89

a) 1

**b) 2**

c) 7

d) 6

5. Which collision resolution technique involves using a hash function to probe a sequence of buckets in a quadratic manner?

A) Separate Chaining

B) Linear Probing

**C) Quadratic Probing**

D) Double Hashing

6. What are the values of h1(k) and h2(k) in the hash function?

a)

h1(k) = m mod k

h2(k) = 1+ (m’ mod k)

b)

h1(k) = 1 + (m mod k)

h2(k) = m’ mod k

c)

h1(k) = 1+ (k mod m)

h2(k) = k mod m

**d)**

**h1(k) = k mod m**

**h2(k) = 1+ (k mod m’)**

7. In double hashing, how is the step size (increment) determined for probing?

A) It is fixed for all keys.

B) It is the same as the hash code of the key.

**C) It is calculated using a secondary hash function.**

D) It is determined by the size of the hash table.

8. At what position the number 72 gets inserted in the following table?

Index Key

0 22

1 34

2

3

4

5 56

6

7 18

8 41

9

10

a) 3

b) 10

c) 4

**d) 6**

9. Which collision resolution technique aims to minimize clustering by spreading out the probing sequence?

A) Separate Chaining

B) Linear Probing

C) Quadratic Probing

**D) Double Hashing**

10. Where does the number 14 get inserted in the following table?

Index Key

0

1 79

2

3

4 69

5 98

6

7 72

8

9

10

11 50

12

a) 2

**b) 9**

c) 4

d) 8

11. Which collision resolution technique tends to have more efficient insertion and deletion operations when the load factor is low?

**A) Separate Chaining**

B) Linear Probing

C) Quadratic Probing

D) Double Hashing

12. What the function of give code snippet?

bool func(int arr1[], int m, int arr2[], int n)

{

set<int> hashset;

for (int i = 0; i < m; i++) {

hashset.insert(arr1[i]);

}

for (int i = 0; i < n; i++) {

if (hashset.find(arr2[i]) == hashset.end())

return false;

}

return true;

}

A) Check if two arrays are equal

**B) Check if arr2 is subset of arr1**

C) Check if arr1 is subset of arr2

D) Merge two arrays

13. Which collision resolution technique may suffer from primary clustering?

A) Separate Chaining

**B) Linear Probing**

C) Quadratic Probing

D) Double Hashing

14.

int inline hash1(int value){

return value%TABLE\_SIZE;

}

int inline hash2(int value){

return PRIME - (value%PRIME);

}

Above functions are used in which type of hashing?

A) Linear probing

**B) Double hashing**

C) Quadratic probing

D) Chaining

15. In separate chaining, what is the primary advantage over other techniques?

A) Reduced memory usage

**B) Fast lookup for all load factors**

C) Efficient insertions and deletions

D) Reduced likelihood of collisions

16. What will be returned by give function in hashing?

float func()

{

return (float)(this->numOfElements + 1) / (float)(this->capacity);

}

A) Number of elements in hash table

B) Capacity of hash table

**C) Load factor**

D) Hash key

17. Which collision resolution technique is less susceptible to secondary clustering?

A) Separate Chaining

B) Linear Probing

C) Quadratic Probing

**D) Double Hashing**

18. What will be the function of below code snippet?

int bucketIndex = this->hashFunction(key);

node<T> \*bucketHead = this->arr[bucketIndex];

while (bucketHead != NULL)

{

if (bucketHead->key == key)

{

return bucketHead->value;

}

bucketHead = bucketHead->next;

}

return -1;

A) Insert key in hash table

B) Delete key from hash table

C) Generate hash value for key

**D) Search key in hash table**

19. In quadratic probing, how is the probing sequence determined for a key with multiple collisions?

A) By adding a fixed offset to the hash code

B) By adding a linear increment to the hash code

**C) By adding a quadratic increment to the hash code**

D) By multiplying the hash code by a fixed factor

20. What will be the function of below code snippet?

int bucketIndex = hashFunction(mp, key);

struct node\* prevNode = NULL;

struct node\* currNode = mp->arr[bucketIndex];

while (currNode != NULL) {

if (strcmp(key, currNode->key) == 0) {

if (currNode == mp->arr[bucketIndex]) {

mp->arr[bucketIndex] = currNode->next;

}

else {

prevNode->next = currNode->next;

}

free(currNode);

break;

}

prevNode = currNode;

currNode = currNode->next;

}

A) Insert key in hash table

**B) Delete key from hash table**

C) Generate hash value for key

D) Search key in hash table