## Marking Rubric

Criteria	Maximum Points
1. Hash Function	
1.1 Correct computation of hash index	4 points
1.2 Handling short keys with given character	3 points
2 .Insert Method	
2.1 Correct insertion for new keys	3 points
2.1 Correct insertion for new keys	4 points
General Functionality & Code     Structure	1 points

## Solution

```
Python
                                                 Java
class Node:
                                                 class Node {
       def init (self, key, value,
                                                        String key;
next=None):
                                                        String value;
       self.key = key
                                                        Node next;
       self.value = value
       self.next = next
                                                        public Node(String key, String value) {
                                                        this.key = key;
                                                        this.value = value;
class HashTable:
       def init (self, size):
                                                        this.next = null;
       self.size = size
                                                        }
       self.table = [None] * size
                                                 class HashTable {
       def hash_function(self, key):
       # Compute hash index based on the
                                                        private Node[] table;
sum of ASCII values of the first three
                                                        private int size;
characters
       ascii_sum = 0
                                                        public HashTable(int size) {
       for i in range(3):
                                                        this.size = size;
       if i < len(key):
                                                        this.table = new Node[size];
               ascii_sum += ord(key[i])
       else:
               ascii_sum += ord('X') # Add
                                                        public int hashFunction(String key) {
'X' if the key is shorter than three characters
                                                        // Compute hash index based on the
       return ascii_sum % self.size
                                                 sum of ASCII values of the first three
```

```
characters
       def insert(self, key, value):
                                                          int asciiSum = 0;
       index = self.hash function(key)
                                                          for (int i = 0; i < 3; i++) {
       new node = Node(key, value)
                                                          if (i < key.length()) {</pre>
                                                                  asciiSum += key.charAt(i);
       if self.table[index] is None:
                                                          } else {
       # No collision, directly insert
                                                                  asciiSum += 'X'; // Add 'X' if
       self.table[index] = new node
                                                  the key is shorter than three characters
       else:
       # Collision resolution with linked list
       current = self.table[index]
                                                          return asciiSum % size:
       while current:
               if current.key == key:
                                                          public void insert(String key, String
               # Update the value if key
already exists
                                                  value) {
               current.value = value
                                                          int index = hashFunction(key);
                                                          Node newNode = new Node(key.
               return
               if current.next is None:
                                                  value);
               # Add new node to the end of
the list
                                                          if (table[index] == null) {
                                                          // No collision, directly insert
               current.next = new node
                                                          table[index] = newNode;
               return
               current = current.next
                                                          } else {
                                                          // Collision resolution with linked list
                                                          Node current = table[index]:
# Example usage
                                                          while (current != null) {
ht = HashTable(10)
                                                                 if (current.key.equals(key)) {
                                                                 // Update the value if key
# Insertions
                                                  already exists
ht.insert("PKG123", "In Transit")
                                                                 current.value = value;
ht.insert("AB", "Delivered")
                                                                  return:
ht.insert("PKG456", "Returned")
                                                                  if (current.next == null) {
                                                                  // Add new node to the end of
print("\nHash table after insertions:")
ht.display()
                                                  the list
                                                                  current.next = newNode;
# Update existing key
                                                                  return;
ht.insert("PKG123", "Delivered")
                                                                 }
                                                                 current = current.next;
print("\nHash table after updates:")
ht.display()
                                                  public static void main(String[] args) {
                                                          // Example usage
                                                          HashTable ht = new HashTable(10);
                                                          // Insertions
                                                          ht.insert("PKG123", "In Transit");
```

```
ht.insert("AB", "Delivered");
ht.insert("PKG456", "Returned");

System.out.println("\nHash table after insertions:");
ht.display();

// Update existing key
ht.insert("PKG123", "Delivered");

System.out.println("\nHash table after updates:");
ht.display();
}

ht.display();
}
```

## SET-B

```
Python
                                                 Java
class Node:
       def init (self, key, value,
                                                 class Node {
next=None):
                                                        String key;
       self.key = key
                                                        double value;
       self.value = value
                                                        Node next; // Pointer to the next node
       self.next = next
                                                 in case of a collision
class HashTable:
                                                        // Constructor for the Node class
       def __init__(self, size):
                                                        public Node(String key, double value,
       self.size = size
                                                 Node next) {
       self.table = [None] * size
                                                        this.key = key;
                                                        this.value = value;
       def hash_function(self, key):
                                                        this.next = next;
       # Compute the hash index based on
the sum of ASCII values of the first three
                                                 }
characters
       ascii sum = 0
                                                 class HashTable {
       for i in range(3):
                                                        private Node[] table; // Array of Node
       if i < len(key):
               ascii_sum += ord(key[i])
                                                 references to store the data
```

```
else:
                                                          private int size; // Size of the hash
               ascii sum += ord('0') # Add '0'
                                                  table
if the key is shorter than three characters
       return ascii sum % self.size
                                                          // Constructor to initialize the table
                                                  with the given size
       def insert(self, key, value):
                                                          public HashTable(int size) {
       index = self.hash function(key)
                                                          this.size = size;
       new node = Node(key, value)
                                                          this.table = new Node[size];
       if self.table[index] is None:
       # No collision, directly insert
       self.table[index] = new node
                                                          public int hash function(String key) {
       else:
                                                          int sum = 0;
       # Collision resolution with linked list
                                                          // Sum the ASCII values of the first
       current = self.table[index]
                                                  three characters
       while current:
                                                          for (int i = 0; i < 3; i++) {
               if current.key == key:
                                                          if (i < key.length()) {
               # Update the value by adding
                                                                 sum += key.charAt(i);
                                                                                                }
the new price to the existing one
                                                  else {
               current.value += value
                                                                 sum += '0'; // If key length is
                                                  less than 3. add ASCII value of '0'
               return
               if current.next is None:
               # Add new node to the end of
the list
                                                          return sum % size; // Return the index
                                                  based on the table size
               current.next = new node
               return
               current = current.next
       def display(self):
                                                          public void insert(String key, double
       for i, node in enumerate(self.table):
                                                  value) {
       if node is None:
                                                          int index = hash function(key); // Get
               continue # Skip empty indices
                                                  the index using the hash function
       print(f"Index {i}:")
                                                          Node newNode = new Node(key,
       current = node
                                                  value, null);
       while current:
               print(f" {current.key}
                                                          // If there's no node at the index, insert
({current.value:.2f})")
                                                  the new node
                                                          if (table[index] == null) {
               current = current.next
                                                          table[index] = newNode;
# Example usage
                                                          } else {
ht = HashTable(10)
                                                          // If collision occurs, traverse the
                                                  linked list
# Insertions
                                                          Node current = table[index];
ht.insert("P123", 19.99)
                                                          while (current != null) {
ht.insert("AB", 15.50)
                                                                 // If the key already exists,
ht.insert("P456", 25.75)
                                                  update the value by adding the new price
                                                                  if (current.kev.equals(kev)) {
print("\nHash table after insertions:")
                                                                  current.value += value;
ht.display()
                                                                  return:
                                                                 }
```

```
# Update existing key
                                                                if (current.next == null) {
                                                                // If we've reached the end of
ht.insert("P123", 21.99)
                                                 the list, add the new node
print("\nHash table after updates:")
                                                                current.next = newNode;
ht.display()
                                                                return;
                                                                current = current.next;
                                                         // Main method to test the hash table
                                                 functionality
                                                         public static void main(String[] args) {
                                                         // Create a new hash table with size
                                                 10
                                                         HashTable ht = new HashTable(10);
                                                         // Insert some sample key-value pairs
                                                         ht.insert("P123", 19.99);
                                                         ht.insert("AB", 15.50);
                                                         ht.insert("P456", 25.75);
                                                         System.out.println("\nHash table after
                                                 insertions:");
                                                         ht.display(); // Display the hash table
                                                 after insertions
                                                         // Update the price for P123 by adding
                                                 the new price
                                                         ht.insert("P123", 21.99);
                                                         System.out.println("\nHash table after
                                                 update:");
                                                         ht.display(); // Display the hash table
                                                 after update
                                                         }
                                                 }
```