| | |
|---|---|
| **Trạng thái** | Đã xong |
| **Bắt đầu vào lúc** | Thứ Ba, 25 tháng 3 2025, 2:39 PM |
| **Kết thúc lúc** | Thứ Ba, 25 tháng 3 2025, 3:39 PM |
| **Thời gian thực hiện** | 1 giờ |
| **Điểm** | 6,00/6,00 |
| **Điểm** | **10,00** trên 10,00 (**100**%) |

Câu hỏi 1

Đúng

Đạt điểm 1,00 trên 1,00

Implement the **put** method in template class **XHashMap** representing the **Hash Table.** The Hash Table is implemented with **Open Hashing** for handling collision, using a Singly linked list to store keys with the same index. The description of the method is given in the code.

```
int hashFunction(int key, int capacity) {
  return key % capacity;
}
template<class K, class V>
class XHashMap {
public:
  class Entry {
  public:
    K key;
    V value;
    Entry* next;

    Entry(K key, V value, Entry* next = 0) {
      this->key = key;
      this->value = value;
      this->next = next;
    }
  };
private:
  Entry** table; // hash table
  int capacity; // size for the hash table
  int count;
public:
  // Constructor
  XHashMap() {
    this->capacity = 10;
    this->count = 0;
    table = new Entry*[capacity];
    //reset table to 0
    for (int i = 0; i < capacity; i++) {
      table[i] = 0;
    }
  }
  ~XHashMap(){ // Destructor
    //Remove all entries in the current map
    for(int idx=0; idx < this->capacity; idx++){
      Entry* entry = this->table[idx];
      while(entry != 0){
        Entry * next = entry->next;
        delete entry;
        entry = next;
      }
    }
    //Remove table
    delete []table;
  }
  // YOUR ANSWER
  // print table method (for testing)
  void printTable() const {
    cout << "------------The hash table is shown below------------\n";
    for (int i = 0; i < capacity; i++) {
      Entry* entry = table[i];
```

```
            cout << "Index " << i << ": ";
            while (entry != 0) {
                cout << "(" << entry->key << ", " << entry->value << ") ";
                entry = entry->next;
            }
            cout << endl;
        }
        cout << "---------------------------------------------------\n";

    }
};
```

**For example:**

| Test | Result |
|---|---|
| map.put(6, 100);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6: (6, 100)<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- |
| map.put(0, 10);<br>map.put(10, 100); // adding not replacing<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0: (10, 100) (0, 10)<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- |
| map.put(1, 15);<br>map.put(1, 200); // updating new value for existing index<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (1, 200)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- |

**Answer:** (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
void put(int key, int value) {
  // Compute the index using the hash function
    int index = hashFunction(key, capacity);

    // Check if the key exists in the linked list at the index
    Entry* current = table[index];
    while (current != 0) {
        // If key is found, update its value
        if (current->key == key) {
            current->value = value;
            return;
        }
        current = current->next;
    }

    // If key doesn't exist, create and insert a new entry
    Entry* newEntry = new Entry(key, value, table[index]);
    table[index] = newEntry;

    // Increment the count and ensure load factor
    count++;
    ensureLoadFactor(count);

}

void ensureLoadFactor(int current_size) {
    // Calculate the maximum allowed size based on the load factor (0.75 * capacity)
    int maxAllowedSize = 0.75 * capacity;

    // If the current size exceeds or equals the max allowed size, trigger rehashing
    if (current_size >= maxAllowedSize) {
        // Calculate the new capacity (1.5 times the old capacity)
        int newCapacity = capacity * 1.5;

        // Call the rehash function with the new capacity
        rehash(newCapacity);
    }
}

void rehash(int newCapacity) {
  // Store the current table and capacity
    Entry** oldTable = table;
    int oldCapacity = capacity;

    // Create a new table with the new capacity and update the capacity
    capacity = newCapacity;
    table = new Entry*[newCapacity];

    // Initialize the new table with nullptr values
    for (int i = 0; i < newCapacity; i++) {
        table[i] = 0;
    }

    // Reset count as we'll be reinserting all entries
    count = 0;

    // For each index in the old table
    for (int i = 0; i < oldCapacity; i++) {
        Entry* current = oldTable[i];

        // Traverse the linked list at that index
        while (current != 0) {
```

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | map.put(6, 100);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6: (6, 100)<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6: (6, 100)<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | map.put(0, 10);<br>map.put(10, 100); // adding not replacing<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0: (10, 100) (0, 10)<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0: (10, 100) (0, 10)<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(1, 15);<br>map.put(1, 200); // updating new value for existing index<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (1, 200)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (1, 200)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(101, 500);<br>map.put(101, 100);<br>map.put(101, 300);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (101, 300)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (101, 300)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(874, 912);<br>map.put(557, 367);<br>map.put(738, 612);<br>map.put(986, 477);<br>map.put(424, 315);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4: (424, 315) (874, 912)<br>Index 5:<br>Index 6: (986, 477)<br>Index 7: (557, 367)<br>Index 8: (738, 612)<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3:<br>Index 4: (424, 315) (874, 912)<br>Index 5:<br>Index 6: (986, 477)<br>Index 7: (557, 367)<br>Index 8: (738, 612)<br>Index 9:<br>----------------------------------------------------- | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | map.put(112, 545);<br>map.put(790, 999);<br>map.put(350, 432);<br>map.put(678, 805);<br>map.put(432, 217);<br>map.put(883, 374);<br>map.put(112, 596);<br>map.put(926, 314);<br>map.put(240, 890);<br>map.put(432, 410);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0: (240, 890)<br>Index 1:<br>Index 2:<br>Index 3: (678, 805)<br>Index 4:<br>Index 5: (350, 432)<br>Index 6:<br>Index 7: (112, 596)<br>Index 8:<br>Index 9:<br>Index 10: (790, 999)<br>Index 11: (926, 314)<br>Index 12: (432, 410)<br>Index 13: (883, 374)<br>Index 14:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0: (240, 890)<br>Index 1:<br>Index 2:<br>Index 3: (678, 805)<br>Index 4:<br>Index 5: (350, 432)<br>Index 6:<br>Index 7: (112, 596)<br>Index 8:<br>Index 9:<br>Index 10: (790, 999)<br>Index 11: (926, 314)<br>Index 12: (432, 410)<br>Index 13: (883, 374)<br>Index 14:<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(510, 789);<br>map.put(734, 645);<br>map.put(510, 132);<br>map.put(341, 981);<br>map.put(210, 473);<br>map.put(653, 550);<br>map.put(480, 280);<br>map.put(556, 990);<br>map.put(808, 359);<br>map.put(510, 732);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0: (510, 732) (210, 473) (480, 280)<br>Index 1: (556, 990)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8: (653, 550)<br>Index 9:<br>Index 10:<br>Index 11: (341, 981)<br>Index 12:<br>Index 13: (808, 359)<br>Index 14: (734, 645)<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0: (510, 732) (210, 473) (480, 280)<br>Index 1: (556, 990)<br>Index 2:<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8: (653, 550)<br>Index 9:<br>Index 10:<br>Index 11: (341, 981)<br>Index 12:<br>Index 13: (808, 359)<br>Index 14: (734, 645)<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(601, 238);<br>map.put(148, 870);<br>map.put(481, 711);<br>map.put(753, 922);<br>map.put(642, 583);<br>map.put(191, 774);<br>map.put(854, 417);<br>map.put(770, 101);<br>map.put(980, 346);<br>map.put(529, 908);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (601, 238) (481, 711)<br>Index 2:<br>Index 3: (753, 922)<br>Index 4: (529, 908)<br>Index 5: (980, 346) (770, 101)<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>Index 10:<br>Index 11: (191, 774)<br>Index 12: (642, 583)<br>Index 13: (148, 870)<br>Index 14: (854, 417)<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1: (601, 238) (481, 711)<br>Index 2:<br>Index 3: (753, 922)<br>Index 4: (529, 908)<br>Index 5: (980, 346) (770, 101)<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>Index 10:<br>Index 11: (191, 774)<br>Index 12: (642, 583)<br>Index 13: (148, 870)<br>Index 14: (854, 417)<br>----------------------------------------------------- | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | map.put(143, 541);<br>map.put(143, 222);<br>map.put(143, 735);<br>map.put(143, 900);<br>map.put(143, 587);<br>map.put(143, 101);<br>map.put(143, 348);<br>map.put(143, 641);<br>map.put(143, 924);<br>map.put(143, 541);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3: (143, 541)<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2:<br>Index 3: (143, 541)<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7:<br>Index 8:<br>Index 9:<br>----------------------------------------------------- | ✓ |
| ✓ | map.put(132, 664);<br>map.put(312, 305);<br>map.put(232, 743);<br>map.put(322, 101);<br>map.put(452, 651);<br>map.put(542, 836);<br>map.put(672, 129);<br>map.put(762, 432);<br>map.put(892, 923);<br>map.put(982, 489);<br>map.printTable(); | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2: (452, 651) (542, 836)<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7: (982, 489) (892, 923) (232, 743) (322, 101)<br>Index 8:<br>Index 9:<br>Index 10:<br>Index 11:<br>Index 12: (762, 432) (132, 664) (312, 305) (672, 129)<br>Index 13:<br>Index 14:<br>----------------------------------------------------- | ------------The hash table is shown below------------<br>Index 0:<br>Index 1:<br>Index 2: (452, 651) (542, 836)<br>Index 3:<br>Index 4:<br>Index 5:<br>Index 6:<br>Index 7: (982, 489) (892, 923) (232, 743) (322, 101)<br>Index 8:<br>Index 9:<br>Index 10:<br>Index 11:<br>Index 12: (762, 432) (132, 664) (312, 305) (672, 129)<br>Index 13:<br>Index 14:<br>----------------------------------------------------- | ✓ |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **2**

Đúng

Đạt điểm 1,00 trên 1,00

Implement the **get** method in template class **XHashMap** representing the **Hash Table.** The Hash Table is implemented with **Open Hashing** for handling collision, using a Singly linked list to store keys with the same index. The description of the method is given in the code.

```
int hashFunction(int key, int capacity) {
    return key % capacity;
}
template<class K, class V>
class XHashMap {
public:
    class Entry {
    public:
        K key;
        V value;
        Entry* next;

        Entry(K key, V value, Entry* next = 0) {
            this->key = key;
            this->value = value;
            this->next = next;
        }
    };
private:
    Entry** table; // hash table
    int capacity; // size for the hash table
    int count;
public:
    // Constructor
    XHashMap() {
        this->capacity = 10;
        this->count = 0;
        table = new Entry*[capacity];
        //reset table to 0
        for (int i = 0; i < capacity; i++) {
            table[i] = 0;
        }
    }
    ~XHashMap(){ // Destructor
        //Remove all entries in the current map
        for(int idx=0; idx < this->capacity; idx++){
            Entry* entry = this->table[idx];
            while(entry != 0){
                Entry * next = entry->next;
                delete entry;
                entry = next;
            }
        }
        //Remove table
        delete []table;
    }

    // put method
    void put(int key, int value); //Already implemented

    // YOUR ANSWER
};
```

**For example:**

| Test | Result |
|---|---|
| vector<int> keys = {20};<br>for(int key: keys){<br>    try{<br>        cout << "Value for key " << key <<": " << map.get(key) << endl;<br>    }<br>    catch (std::out_of_range& e){<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 20: 95 |

**Answer:** (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
V& get(K key) {
    // Calculate the index using the hash function
    int index = hashFunction(key, capacity);

    // Access and traverse the linked list at that index
    Entry* current = table[index];
    while (current != 0) {
        // If the current entry's key matches, return its value
        if (current->key == key) {
            return current->value;
        }
        current = current->next;
    }

    // If not found, throw std::out_of_range exception
    throw std::out_of_range("Key not found");
}
```

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {20};<br>for(int key: keys){<br>    try{<br>        cout << "Value for key " << key <<": " << map.get(key) << endl;<br>    }<br>    catch (std::out_of_range& e){<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 20: 95 | Value for key 20: 95 | ✓ |
| ✓ | vector<int> keys = {52, 35};<br><br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (const std::out_of_range& e) {<br>        cout << "Key " << key << " not found: " << e.what() << endl;<br>    }<br>} | Value for key 52: 70<br>Value for key 35: 99 | Value for key 52: 70<br>Value for key 35: 99 | ✓ |
| ✓ | vector<int> keys = {20, 1, 57};<br>for(int key: keys){<br>    try{<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    }<br>    catch (std::out_of_range& e){<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 20: 95<br>Value for key 1: Key not found<br>Value for key 57: 80 | Value for key 20: 95<br>Value for key 1: Key not found<br>Value for key 57: 80 | ✓ |
| ✓ | vector<int> keys = {87, 65, 43, 92, 12};<br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (std::out_of_range& e) {<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 87: 77<br>Value for key 65: Key not found<br>Value for key 43: 15<br>Value for key 92: 32<br>Value for key 12: Key not found | Value for key 87: 77<br>Value for key 65: Key not found<br>Value for key 43: 15<br>Value for key 92: 32<br>Value for key 12: Key not found | ✓ |
| ✓ | vector<int> keys = {53, 91, 79, 4, 99};<br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (std::out_of_range& e) {<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 53: 26<br>Value for key 91: Key not found<br>Value for key 79: Key not found<br>Value for key 4: 74<br>Value for key 99: Key not found | Value for key 53: 26<br>Value for key 91: Key not found<br>Value for key 79: Key not found<br>Value for key 4: 74<br>Value for key 99: Key not found | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {77, 33, 56, 20, 60, 70, 15, 48, 68, 11};<br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (std::out_of_range& e) {<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 77: Key not found<br>Value for key 33: 96<br>Value for key 56: 29<br>Value for key 20: 95<br>Value for key 60: Key not found<br>Value for key 70: 43<br>Value for key 15: Key not found<br>Value for key 48: 31<br>Value for key 68: 57<br>Value for key 11: 72 | Value for key 77: Key not found<br>Value for key 33: 96<br>Value for key 56: 29<br>Value for key 20: 95<br>Value for key 60: Key not found<br>Value for key 70: 43<br>Value for key 15: Key not found<br>Value for key 48: 31<br>Value for key 68: 57<br>Value for key 11: 72 | ✓ |
| ✓ | vector<int> keys = {62, 89, 1, 96, 81, 40, 54, 9, 82, 18};<br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (std::out_of_range& e) {<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 62: Key not found<br>Value for key 89: Key not found<br>Value for key 1: Key not found<br>Value for key 96: Key not found<br>Value for key 81: Key not found<br>Value for key 40: 28<br>Value for key 54: Key not found<br>Value for key 9: 81<br>Value for key 82: 65<br>Value for key 18: 54 | Value for key 62: Key not found<br>Value for key 89: Key not found<br>Value for key 1: Key not found<br>Value for key 96: Key not found<br>Value for key 81: Key not found<br>Value for key 40: 28<br>Value for key 54: Key not found<br>Value for key 9: 81<br>Value for key 82: 65<br>Value for key 18: 54 | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | `vector<int> keys = {94, 71, 26, 25, 84, 97, 17, 100, 31, 42};`<br>`for (int key : keys) {`<br>`    try {`<br>`        cout << "Value for key " << key << ": " << map.get(key) << endl;`<br>`    } catch (std::out_of_range& e) {`<br>`        cout << e.what() << endl;`<br>`    }`<br>`}` | Value for key 94: Key not found<br>Value for key 71: 90<br>Value for key 26: 79<br>Value for key 25: 11<br>Value for key 84: Key not found<br>Value for key 97: 88<br>Value for key 17: 23<br>Value for key 100: Key not found<br>Value for key 31: 45<br>Value for key 42: Key not found | Value for key 94: Key not found<br>Value for key 71: 90<br>Value for key 26: 79<br>Value for key 25: 11<br>Value for key 84: Key not found<br>Value for key 97: 88<br>Value for key 17: 23<br>Value for key 100: Key not found<br>Value for key 31: 45<br>Value for key 42: Key not found | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | `vector<int> keys = {86, 2, 22, 75, 78, 67, 29, 88, 45, 63, 69, 7, 95, 66, 34, 52, 13, 58, 46, 49};`<br>`for (int key : keys) {`<br>`    try {`<br>`        cout << "Value for key " << key << ": " << map.get(key) << endl;`<br>`    } catch (std::out_of_range& e) {`<br>`        cout << e.what() << endl;`<br>`    }`<br>`}` | Value for key 86: Key not found<br>Value for key 2: Key not found<br>Value for key 22: 89<br>Value for key 75: Key not found<br>Value for key 78: 19<br>Value for key 67: Key not found<br>Value for key 29: 48<br>Value for key 88: 68<br>Value for key 45: Key not found<br>Value for key 63: 14<br>Value for key 69: 55<br>Value for key 7: Key not found<br>Value for key 95: Key not found<br>Value for key 66: 85<br>Value for key 34: 86<br>Value for key 52: 70<br>Value for key 13: 84<br>Value for key 58: Key not found<br>Value for key 46: 73<br>Value for key 49: 93 | Value for key 86: Key not found<br>Value for key 2: Key not found<br>Value for key 22: 89<br>Value for key 75: Key not found<br>Value for key 78: 19<br>Value for key 67: Key not found<br>Value for key 29: 48<br>Value for key 88: 68<br>Value for key 45: Key not found<br>Value for key 63: 14<br>Value for key 69: 55<br>Value for key 7: Key not found<br>Value for key 95: Key not found<br>Value for key 66: 85<br>Value for key 34: 86<br>Value for key 52: 70<br>Value for key 13: 84<br>Value for key 58: Key not found<br>Value for key 46: 73<br>Value for key 49: 93 | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {85, 32, 50, 83, 21, 600, 3, 73, 93, 74, 345, 57, 10, 24, 200, 44, 18, 72, 80, 105, 51, 889, 26, 123, 59, 0, 101, 57, 35, 110};<br>for (int key : keys) {<br>    try {<br>        cout << "Value for key " << key << ": " << map.get(key) << endl;<br>    } catch (std::out_of_range& e) {<br>        cout << e.what() << endl;<br>    }<br>} | Value for key 85: Key not found<br>Value for key 32: Key not found<br>Value for key 50: 21<br>Value for key 83: 87<br>Value for key 21: 94<br>Value for key 600: Key not found<br>Value for key 3: 51<br>Value for key 73: 30<br>Value for key 93: 53<br>Value for key 74: Key not found<br>Value for key 345: Key not found<br>Value for key 57: 80<br>Value for key 10: Key not found<br>Value for key 24: Key not found<br>Value for key 200: Key not found<br>Value for key 44: 16<br>Value for key 18: 54<br>Value for key 72: Key not found<br>Value for key 80: Key not found<br>Value for key 105: Key not found<br>Value for key 51: Key not found<br>Value for key 889: Key not found<br>Value for key 26: 79<br>Value for key 123: Key not found<br>Value for key 59: 60<br>Value for key 0: Key not found<br>Value for key 101: Key not found<br>Value for key 57: 80<br>Value for key 35: 99<br>Value for key 110: Key not found | Value for key 85: Key not found<br>Value for key 32: Key not found<br>Value for key 50: 21<br>Value for key 83: 87<br>Value for key 21: 94<br>Value for key 600: Key not found<br>Value for key 3: 51<br>Value for key 73: 30<br>Value for key 93: 53<br>Value for key 74: Key not found<br>Value for key 345: Key not found<br>Value for key 57: 80<br>Value for key 10: Key not found<br>Value for key 24: Key not found<br>Value for key 200: Key not found<br>Value for key 44: 16<br>Value for key 18: 54<br>Value for key 72: Key not found<br>Value for key 80: Key not found<br>Value for key 105: Key not found<br>Value for key 51: Key not found<br>Value for key 889: Key not found<br>Value for key 26: 79<br>Value for key 123: Key not found<br>Value for key 59: 60<br>Value for key 0: Key not found<br>Value for key 101: Key not found<br>Value for key | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| | | | 57: 80<br>Value for key<br>35: 99<br>Value for key<br>110: Key not<br>found | |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **3**

Đúng

Đạt điểm 1,00 trên 1,00

Implement the **remove** method in template class **XHashMap** representing the **Hash Table.** The Hash Table is implemented with **Open Hashing** for handling collision, using a Singly linked list to store keys with the same index. The description of the method is given in the code.

```
int hashFunction(int key, int capacity) {
    return key % capacity;
}
template<class K, class V>
class XHashMap {
public:
    class Entry {
    public:
        K key;
        V value;
        Entry* next;

        Entry(K key, V value, Entry* next = 0) {
            this->key = key;
            this->value = value;
            this->next = next;
        }
    };
private:
    Entry** table; // hash table
    int capacity; // size for the hash table
    int count;
public:
    // Constructor
    XHashMap() {
        this->capacity = 10;
        this->count = 0;
        table = new Entry*[capacity];
        //reset table to 0
        for (int i = 0; i < capacity; i++) {
            table[i] = 0;
        }
    }
    ~XHashMap(){ // Destructor
        //Remove all entries in the current map
        for(int idx=0; idx < this->capacity; idx++){
            Entry* entry = this->table[idx];
            while(entry != 0){
                Entry * next = entry->next;
                delete entry;
                entry = next;
            }
        }
        //Remove table
        delete []table;
    }

    // put method
    void put(int key, int value); //Already implemented
    V& get (int key); //Already implemented

    // YOUR ANSWER
};
```

**For example:**

<table>
<tr><th>Test</th><th>Result</th></tr>
<tr><td><pre>
vector&lt;int&gt; keys = {68};  // Update the keys vector to hold the new key
for (int key : keys) {
    try {
        cout << "Remove for key = " << key << ", value = " << map.remove(key) <<
endl;
    }
    catch (std::out_of_range& e) {  // Catch the out_of_range exception
        cout << e.what() << endl;   // Print the exception message
    }
}
</pre></td><td><pre>
Remove for key = 68, value = 57
</pre></td></tr>
<tr><td><pre>
vector&lt;int&gt; keys = {92, 51, 34};  // Include all keys in the vector
for (int key : keys) {
    try {
        cout << "Remove for key = " << key << ", value = " << map.remove(key) <<
endl;
    }
    catch (std::out_of_range& e) {  // Catch the out_of_range exception
        cout << e.what() << endl;   // Print the exception message
    }
}
</pre></td><td><pre>
Remove for key = 92, value = 32
Remove for key = 51, value = Key not
found
Remove for key = 34, value = 86
</pre></td></tr>
<tr><td><pre>
vector&lt;int&gt; keys = {83, 4, 77, 28, 56};  // Update the keys vector with the new
keys
for (int key : keys) {
    try {
        cout << "Remove for key = " << key << ", value = " << map.remove(key) <<
endl;
    }
    catch (std::out_of_range& e) {  // Catch the out_of_range exception
        cout << e.what() << endl;   // Print the exception message
    }
}
</pre></td><td><pre>
Remove for key = 83, value = 87
Remove for key = 4, value = 74
Remove for key = 77, value = Key not
found
Remove for key = 28, value = Key not
found
Remove for key = 56, value = 29
</pre></td></tr>
</table>

**Answer:** (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
V remove(K key) {
    // Calculate the index using the hash function
    int index = hashFunction(key, capacity);

    // Start at the head of the linked list for this index
    Entry* current = table[index];
    Entry* prev = nullptr;

    // Traverse the linked list to find the key
    while (current != nullptr) {
        // If key is found
        if (current->key == key) {
            // Store the value to return
            V value = current->value;

            // If this is the first node in the list
            if (prev == nullptr) {
                // Update the head of the list
                table[index] = current->next;
            }
            else {
                // Link previous node to the next node
                prev->next = current->next;
            }

            // Decrement the count
            count--;

            // Delete the entry and return its value
            delete current;
            return value;
        }

        // Move to next node
        prev = current;
        current = current->next;
    }

    // If key is not found, throw an out_of_range exception
    throw std::out_of_range("Key not found");
}
```

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {68};  // Update the keys vector to hold the new key<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 68, value = 57 | Remove for key = 68, value = 57 | ✓ |
| ✓ | vector<int> keys = {92, 51, 34};  // Include all keys in the vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 92, value = 32<br>Remove for key = 51, value = Key not found<br>Remove for key = 34, value = 86 | Remove for key = 92, value = 32<br>Remove for key = 51, value = Key not found<br>Remove for key = 34, value = 86 | ✓ |
| ✓ | vector<int> keys = {83, 4, 77, 28, 56};  // Update the keys vector with the new keys<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 83, value = 87<br>Remove for key = 4, value = 74<br>Remove for key = 77, value = Key not found<br>Remove for key = 28, value = Key not found<br>Remove for key = 56, value = 29 | Remove for key = 83, value = 87<br>Remove for key = 4, value = 74<br>Remove for key = 77, value = Key not found<br>Remove for key = 28, value = Key not found<br>Remove for key = 56, value = 29 | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {65, 21, 100, 47, 59, 14, 99, 76, 22, 70};  // Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 65, value = Key not found<br>Remove for key = 21, value = 94<br>Remove for key = 100, value = Key not found<br>Remove for key = 47, value = Key not found<br>Remove for key = 59, value = 60<br>Remove for key = 14, value = 23<br>Remove for key = 99, value = Key not found<br>Remove for key = 76, value = 43<br>Remove for key = 22, value = 89<br>Remove for key = 70, value = 43 | Remove for key = 65, value = Key not found<br>Remove for key = 21, value = 94<br>Remove for key = 100, value = Key not found<br>Remove for key = 47, value = Key not found<br>Remove for key = 59, value = 60<br>Remove for key = 14, value = 23<br>Remove for key = 99, value = Key not found<br>Remove for key = 76, value = 43<br>Remove for key = 22, value = 89<br>Remove for key = 70, value = 43 | ✓ |
| ✓ | vector<int> keys = {11, 41, 87, 9, 39, 7, 25, 46, 40, 16};  // Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 11, value = 72<br>Remove for key = 41, value = 62<br>Remove for key = 87, value = 77<br>Remove for key = 9, value = 81<br>Remove for key = 39, value = 91<br>Remove for key = 7, value = Key not found<br>Remove for key = 25, value = 11<br>Remove for key = 46, value = 73<br>Remove for key = 40, value = 28<br>Remove for key = 16, value = Key not found | Remove for key = 11, value = 72<br>Remove for key = 41, value = 62<br>Remove for key = 87, value = 77<br>Remove for key = 9, value = 81<br>Remove for key = 39, value = 91<br>Remove for key = 7, value = Key not found<br>Remove for key = 25, value = 11<br>Remove for key = 46, value = 73<br>Remove for key = 40, value = 28<br>Remove for key = 16, value = Key not found | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {85, 73, 80, 15, 66, 88, 60, 53, 90, 57};<br>// Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 85, value = Key not found<br>Remove for key = 73, value = 30<br>Remove for key = 80, value = Key not found<br>Remove for key = 15, value = Key not found<br>Remove for key = 66, value = 85<br>Remove for key = 88, value = 68<br>Remove for key = 60, value = Key not found<br>Remove for key = 53, value = 26<br>Remove for key = 90, value = Key not found<br>Remove for key = 57, value = 80 | Remove for key = 85, value = Key not found<br>Remove for key = 73, value = 30<br>Remove for key = 80, value = Key not found<br>Remove for key = 15, value = Key not found<br>Remove for key = 66, value = 85<br>Remove for key = 88, value = 68<br>Remove for key = 60, value = Key not found<br>Remove for key = 53, value = 26<br>Remove for key = 90, value = Key not found<br>Remove for key = 57, value = 80 | ✓ |
| ✓ | vector<int> keys = {26, 1, 5, 94, 42, 43, 97, 20, 64, 54};<br>// Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 26, value = 79<br>Remove for key = 1, value = Key not found<br>Remove for key = 5, value = 37<br>Remove for key = 94, value = Key not found<br>Remove for key = 42, value = Key not found<br>Remove for key = 43, value = 15<br>Remove for key = 97, value = 88<br>Remove for key = 20, value = 95<br>Remove for key = 64, value = 49<br>Remove for key = 54, value = Key not found | Remove for key = 26, value = 79<br>Remove for key = 1, value = Key not found<br>Remove for key = 5, value = 37<br>Remove for key = 94, value = Key not found<br>Remove for key = 42, value = Key not found<br>Remove for key = 43, value = 15<br>Remove for key = 97, value = 88<br>Remove for key = 20, value = 95<br>Remove for key = 64, value = 49<br>Remove for key = 54, value = Key not found | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {37, 93, 18, 62, 86, 46, 30, 35, 8, 69, 96, 24, 81, 13, 52, 84, 45, 23, 2, 31};  // Updated keys vector<br>for (int key : keys) {<br>try {<br>cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>}<br>catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>cout << e.what() << endl;   // Print the exception message<br>}<br>} | Remove for key = 37, value = Key not found<br>Remove for key = 93, value = 53<br>Remove for key = 18, value = 54<br>Remove for key = 62, value = Key not found<br>Remove for key = 86, value = Key not found<br>Remove for key = 46, value = 73<br>Remove for key = 30, value = 52<br>Remove for key = 35, value = 99<br>Remove for key = 8, value = 66<br>Remove for key = 69, value = 55<br>Remove for key = 96, value = Key not found<br>Remove for key = 24, value = Key not found<br>Remove for key = 81, value = Key not found<br>Remove for key = 13, value = 84<br>Remove for key = 52, value = 70<br>Remove for key = 84, value = Key not found<br>Remove for key = 45, value = Key not found<br>Remove for key = 23, value = Key not found<br>Remove for key = 2, value = Key not found<br>Remove for key = 31, value = 45 | Remove for key = 37, value = Key not found<br>Remove for key = 93, value = 53<br>Remove for key = 18, value = 54<br>Remove for key = 62, value = Key not found<br>Remove for key = 86, value = Key not found<br>Remove for key = 46, value = 73<br>Remove for key = 30, value = 52<br>Remove for key = 35, value = 99<br>Remove for key = 8, value = 66<br>Remove for key = 69, value = 55<br>Remove for key = 96, value = Key not found<br>Remove for key = 24, value = Key not found<br>Remove for key = 81, value = Key not found<br>Remove for key = 13, value = 84<br>Remove for key = 52, value = 70<br>Remove for key = 84, value = Key not found<br>Remove for key = 45, value = Key not found<br>Remove for key = 23, value = Key not found<br>Remove for key = 2, value = Key not found<br>Remove for key = 31, value = 45 | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {12, 61, 82, 10, 17, 49, 33, 74, 75, 86,<br>                 57, 78, 19, 50, 27, 67, 38, 100, 15, 45};  // Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 12, value = Key not found<br>Remove for key = 61, value = 58<br>Remove for key = 82, value = 65<br>Remove for key = 10, value = Key not found<br>Remove for key = 17, value = 23<br>Remove for key = 49, value = 93<br>Remove for key = 33, value = 96<br>Remove for key = 74, value = Key not found<br>Remove for key = 75, value = Key not found<br>Remove for key = 86, value = Key not found<br>Remove for key = 57, value = 80<br>Remove for key = 78, value = 19<br>Remove for key = 19, value = Key not found<br>Remove for key = 50, value = 21<br>Remove for key = 27, value = Key not found<br>Remove for key = 67, value = Key not found<br>Remove for key = 38, value = Key not found<br>Remove for key = 100, value = Key not found<br>Remove for key = 15, value = Key not found<br>Remove for key = 45, value = Key not found | Remove for key = 12, value = Key not found<br>Remove for key = 61, value = 58<br>Remove for key = 82, value = 65<br>Remove for key = 10, value = Key not found<br>Remove for key = 17, value = 23<br>Remove for key = 49, value = 93<br>Remove for key = 33, value = 96<br>Remove for key = 74, value = Key not found<br>Remove for key = 75, value = Key not found<br>Remove for key = 86, value = Key not found<br>Remove for key = 57, value = 80<br>Remove for key = 78, value = 19<br>Remove for key = 19, value = Key not found<br>Remove for key = 50, value = 21<br>Remove for key = 27, value = Key not found<br>Remove for key = 67, value = Key not found<br>Remove for key = 38, value = Key not found<br>Remove for key = 100, value = Key not found<br>Remove for key = 15, value = Key not found<br>Remove for key = 45, value = Key not found | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | vector<int> keys = {540, 89, 200, 99, 111, 73, 120, 23, 62, 101,<br>                    92, 133, 57, 999, 80, 109, 66, 124, 66, 137,<br>                    321, 106, 95, 128, 63, 118, 102, 55, 87, 1};  // Updated keys vector<br>for (int key : keys) {<br>    try {<br>        cout << "Remove for key = " << key << ", value = " << map.remove(key) << endl;<br>    }<br>    catch (std::out_of_range& e) {  // Catch the out_of_range exception<br>        cout << e.what() << endl;   // Print the exception message<br>    }<br>} | Remove for key = 540, value = Key not found<br>Remove for key = 89, value = Key not found<br>Remove for key = 200, value = Key not found<br>Remove for key = 99, value = Key not found<br>Remove for key = 111, value = Key not found<br>Remove for key = 73, value = 30<br>Remove for key = 120, value = Key not found<br>Remove for key = 23, value = Key not found<br>Remove for key = 62, value = Key not found<br>Remove for key = 101, value = Key not found<br>Remove for key = 92, value = 32<br>Remove for key = 133, value = Key not found<br>Remove for key = 57, value = 80<br>Remove for key = 999, value = Key not found<br>Remove for key = 80, value = Key not found<br>Remove for key = 109, value = Key not found<br>Remove for key = 66, value = 85<br>Remove for key = 124, value = Key not found<br>Remove for key = 66, value = Key not found<br>Remove for key = 137, value = Key not found<br>Remove for key = 321, value = Key not found<br>Remove for key = 106, value = Key not found<br>Remove for key = 95, value = Key not found<br>Remove for key = 128, value = Key not found<br>Remove for key = 63, value = 14<br>Remove for key = 118, value = Key not found<br>Remove for key = 102, value = Key not found<br>Remove for key = 55, value = Key not found<br>Remove for key = 87, value = 77<br>Remove for key = 1, value = Key not found | Remove for key = 540, value = Key not found<br>Remove for key = 89, value = Key not found<br>Remove for key = 200, value = Key not found<br>Remove for key = 99, value = Key not found<br>Remove for key = 111, value = Key not found<br>Remove for key = 73, value = 30<br>Remove for key = 120, value = Key not found<br>Remove for key = 23, value = Key not found<br>Remove for key = 62, value = Key not found<br>Remove for key = 101, value = Key not found<br>Remove for key = 92, value = 32<br>Remove for key = 133, value = Key not found<br>Remove for key = 57, value = 80<br>Remove for key = 999, value = Key not found<br>Remove for key = 80, value = Key not found<br>Remove for key = 109, value = Key not found<br>Remove for key = 66, value = 85<br>Remove for key = 124, value = Key not found<br>Remove for key = 66, value = Key not found<br>Remove for key = 137, value = Key not found<br>Remove for key = 321, value = Key not found<br>Remove for key = 106, value = Key not found<br>Remove for key = 95, value = Key not found<br>Remove for key = 128, value = Key not found<br>Remove for key = 63, value = 14<br>Remove for key = | ✓ |

| | Test | Expected | Got | |
|---|---|---|---|---|
| | | | 118, value = Key not found<br>Remove for key = 102, value = Key not found<br>Remove for key = 55, value = Key not found<br>Remove for key = 87, value = 77<br>Remove for key = 1, value = Key not found | |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **4**

Đúng

Đạt điểm 1,00 trên 1,00

Implement three following hashing function:

```
long int midSquare(long int seed);
long int moduloDivision(long int seed, long int mod);
long int digitExtraction(long int seed, int* extractDigits, int size);
```

Note that:

In midSquare function: we eliminate 2 last digits and get the 4 next digits.

In digitExtraction: extractDigits is a sorted array from smallest to largest index of digit in seed (index starts from 0). The array has size **size.**

**For example:**

| Test | Result |
|---|---|
| `int a[]={1,2,5};`<br>`cout << digitExtraction(122443,a,3);` | 223 |
| `cout <<midSquare(9452);` | 3403 |

**Answer:** (penalty regime: 0, 0, 0 %)

Reset answer

```
long int midSquare(long int seed)
{
    seed = seed * seed;
    seed /= 100;
    return seed % 10000;
}
long int moduloDivision(long int seed, long int mod)
{
    return seed % mod;
}
long int digitExtraction(long int seed,int* extractDigits,int size)
{
   int tmp[1000];
    for(int i = 0;i < 1000; i++) tmp[i] = -1;
    int i = 0;
    while(seed > 0){
        tmp[i] = seed % 10;
        seed /= 10;
        i++;
    }
    long int result = 0;
    long int n = 1;
    int j = 0;
    while(size){
        result = result * 10 + tmp[i - extractDigits[j] - 1];
        j += 1;
        n *= 10;
        size --;

    }
    return result;
}
```

|  | Test | Expected | Got |  |
|---|---|---|---|---|
| ✓ | `int a[]={1,2,5};`<br>`cout << digitExtraction(122443,a,3);` | 223 | 223 | ✓ |
| ✓ | `cout <<midSquare(9452);` | 3403 | 3403 | ✓ |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **5**

Đúng

Đạt điểm 1,00 trên 1,00

Implement function

```
int foldShift(long long key, int addressSize);
int rotation(long long key, int addressSize);
```

to hashing key using Fold shift or Rotation algorithm.

Review:

The **folding method** for constructing hash functions begins by dividing the item into equal-size pieces (the last piece may not be of equal size). These pieces are then added together to give the resulting hash value.

The **rotation** method rotates the last digit to the front, and apply foldShift.

**For example:**

| Test | Result |
|---|---|
| `cout << rotation(600101, 2);` | 26 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
#include<math.h>
#include<string.h>
long int to_int(string s) {
    long int base=1;
    long int res=0;
    for (int i=s.size()-1; i>=0; i--) {
        res += (s[i]-48)*base;
        base*=10;
    }
    return res;
}
int foldShift(long long key, int addressSize)
{
    string s="";
    string num=to_string(key);
    long int sum=0;
    for (int i=0; i<int(num.size()); ) {
        s="";
        for (int j=0; j<addressSize&&i+j<int(num.size()); j++) {
            s+= num[i+j];
        }
        i=i+addressSize;
        sum+=to_int(s);
    }
     long int mod = pow(10, addressSize);
    return sum % mod;
}

int rotation(long long key, int addressSize)
{
    string num=to_string(key);
    string s1 = num.substr(0, num.size()-1);
    num=num[num.size()-1]+s1;
    long int n = to_int(num);
    return foldShift(n, addressSize);
    return 0;
}
```

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | `cout << rotation(600101, 2);` | 26 | 26 | ✓ |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **6**

Đúng

Đạt điểm 1,00 trên 1,00

There are `n` people, each person has a number between `1` and `100000` (`1 ≤ n ≤ 100000`). Given a number `target`. Two people can be matched as a **perfect pair** if the sum of numbers they have is equal to `target`. A person can be matched no more than 1 time.

**Request:** Implement function:

```
int pairMatching(vector<int>& nums, int target);
```

Where `nums` is the list of numbers of `n` people, `target` is the given number. This function returns the number of **perfect pairs** can be found from the list.

**Example:**

The list of numbers is `{1, 3, 5, 3, 7}` and `target = 6`. Therefore, the number of **perfect pairs** can be found from the list is `2` (pair `(1, 5)` and pair `(3, 3)`).

**Note:**

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

**For example:**

| Test | Result |
|---|---|
| `vector<int>items{1, 3, 5, 3, 7};`<br>`int target = 6;`<br>`cout << pairMatching(items, target);` | 2 |
| `int target = 6;`<br>`vector<int>items{4,4,2,1,2};`<br>`cout << pairMatching(items, target);` | 2 |

**Answer:** (penalty regime: 0, 0, 0, 5, 10, ... %)

Reset answer

```
int pairMatching(vector<int>& nums, int target) {
    map<int, int> m;

    // Đếm số lần xuất hiện của từng số
    for(int i : nums)
        m[i]++;

    int count = 0;

    for(auto it = m.begin(); it != m.end(); ++it) {
        int num = it->first;
        int freq = it->second;

        // Trường hợp num + num == target
        if (num * 2 == target) {
            count += freq / 2;
            m[num] = 0; // Đánh dấu là đã sử dụng hết số này
        }
        // Trường hợp tìm thấy cặp (num, target - num)
        else if (m.find(target - num) != m.end() && m[target - num] > 0) {
            int pair_count = min(freq, m[target - num]);
            count += pair_count;
            m[num] = 0;
            m[target - num] = 0; // Đánh dấu cả hai số là đã sử dụng
        }
    }
    return count;
}
```

| | Test | Expected | Got | |
|---|---|---|---|---|
| ✓ | `vector<int>items{1, 3, 5, 3, 7};`<br>`int target = 6;`<br>`cout << pairMatching(items, target);` | 2 | 2 | ✓ |

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.