Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Ba, 16 tháng 4 2024, 1:44 PM
Kết thúc lúc	Thứ Ba, 16 tháng 4 2024, 1:48 PM
Thời gian thực hiện	4 phút 22 giây

Câu hỏi **1**

Đúng

Đạt điểm 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 Fold shift:

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.

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
int foldShift(long long key, int addressSize) {
    string num = to_string(key);
    int sum = 0;
    int size = num.size();
    for (int i = 0; i < size; i += addressSize) {
        string s = num.substr(i, addressSize);
        sum += stoi(s);
    }
    return sum % (int)(pow(10, addressSize));
}

int rotation(long long key, int addressSize) {
    string num = to_string(key);
    char lastChar = num.back();
    num.pop_back();
    num = lastChar + num;
    long long n = stoll(num);
    return foldShift(n, addressSize);
}
```

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

Passed all tests! ✓

Câu hỏi **2**

Đúng

Đạt điểm 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}; 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 a[1000] = {0};
    int i = 0;
    while (seed > 0) {
        a[i] = seed % 10;
        seed /= 10;
        ++i;
    }
    long int result = 0;
    for (int j = 0; j < size; ++j) {
        result = result * 10 + a[i - extractDigits[j] - 1];
    }
```

```
24      return result;
25  }
```

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

Passed all tests! ✓

Câu hỏi **3**

Đúng

Đạt điểm 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};`
`int target = 6;`
`cout << pairMatching(items, target);`
	2

`int target = 6;`
`vector<int>items{4,4,2,1,2};`
`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> h;
    for (int i : nums) {
        ++h[i];
    }
    int cnt = 0;
    for (int i : nums) {
        if (i + i == target) {
            if (h[i] > 1) {
                ++cnt;
                h[i] -= 2;
            }
        }
        else if (h[target - i] > 0 && h[i] > 0){
            ++cnt;
```

```
            --h[i];
            --h[target - i];
        }
    }
    return cnt;
}
```

	Test	Expected	Got	

✓	vector<int>items{1, 3, 5, 3, 7};
	int target = 6;
	cout << pairMatching(items, target);
		2	2	✓

Passed all tests! ✓