Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 18 tháng 9 2024, 8:50 AM
Kết thúc lúc	Thứ Hai, 30 tháng 9 2024, 3:36 PM
Thời gian thực hiện	12 Các ngày 6 giờ
Điểm	6,50/7,00
Điểm	**9,29** trên 10,00 (**92,86**%)

Câu hỏi **1**

Đúng

Đạt điểm 1,00 trên 1,00

Implement methods **ensureCapacity**, **add**, **size** in template class **ArrayList** representing the array list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class ArrayList {
protected:
    T* data;        // dynamic array to store the list's items
    int capacity;   // size of the dynamic array
    int count;      // number of items stored in the array
public:
    ArrayList(){capacity = 5; count = 0; data = new T[5];}
```

```
    ~ArrayList(){ delete[] data; }
    void    add(T e);
    void    add(int index, T e);
    int     size();
    void    ensureCapacity(int index);
};
```

For example:

Test	**Result**

`ArrayList<int> arr;`
`int size = 10;`
 	
`for(int index = 0; index < size; index++){`
`    arr.add(index);`
`}`
 	
`cout << arr.toString() << '\n';`
`cout << arr.size();`
	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
	`10`

`ArrayList<int> arr;`
`int size = 20;`
 	
`for(int index = 0; index < size; index++){`
`    arr.add(0, index);`
`}`
 	
`cout << arr.toString() << '\n';`
`cout << arr.size() << '\n';`
`arr.ensureCapacity(5);`
	`[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]`
	`20`

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

Reset answer

```
        //         CALL resize function to increase the capacity of the array
        //     END IF

        //     SET array[size] to element
        //     INCREMENT size by 1
    //      END FUNCTION
    if (this->count == this->capacity)
    {
        this->ensureCapacity(this->capacity + 1);
    }
    this->data[this->count] = e;
    this->count++;

}

template <class T>
void ArrayList<T>::add(int index, T e)
{
    /*
        FUNCTION add(index, element):
            IF index is less than 0 OR index is greater than count:
                THROW std::out_of_range("the input index is out of range!")
            END IF

            IF count is equal to capacity:
                CALL ensureCapacity with capacity + 1
            END IF

            FOR i from count to index, decrementing by 1:
                SET data[i] to data[i - 1]
            END FOR

            SET data[index] to element
            INCREMENT count by 1
        END FUNCTION
    */
    if (index < 0 || index > this->count)
    {
        throw std::out_of_range("the input index is out of range!");
    }
    if (this->count == this->capacity)
    {
        this->ensureCapacity(this->capacity + 1);
    }
    for (int i = this->count; i > index; i--)
    {
        data[i] = data[i - 1];
    }
    data[index] = e;
    count++;
}

template <class T>
int ArrayList<T>::size()
{
    /*
        FUNCTION size:
            RETURN count
        END FUNCTION
    */
    return this->count;
}
```

	Test	Expected	Got	
✓	`ArrayList<int> arr;` `int size = 10;` `for(int index = 0; index < size; index++){` `arr.add(index);` `}` `cout << arr.toString() << '\n';` `cout << arr.size();`	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]` `10`	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]` `10`	✓

	Test	Expected	Got	

✓	`ArrayList<int> arr;`
	`int size = 20;`
	
	`for(int index = 0; index < size; index++){`
	`    arr.add(0, index);`
	`}`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size() << '\n';`
	`arr.ensureCapacity(5);`
		`[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]`
		`20`
			`[19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]`
			`20`
				✓

Passed all tests! ✓

Đúng

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

Câu hỏi **2**

Đúng

Đạt điểm 1,00 trên 1,00

Implement methods **removeAt, removeItem, clear** in template class **ArrayList** representing the singly linked list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class ArrayList {
```

protected:
T* data; // dynamic array to store the list's items
int capacity; // size of the dynamic array
int count; // number of items stored in the array

```
public:
    ArrayList(){capacity = 5; count = 0; data = new T[5];}
   ~ArrayList(){ delete[] data; }
```

```
    void    add(T e);
    void    add(int index, T e);
    int     size();
    bool    empty();
    void    clear();
    T       get(int index);
    void    set(int index, T e);
    int     indexOf(T item);
    bool    contains(T item);
    T       removeAt(int index);
    bool    removeItem(T item);
```

```
    void    ensureCapacity(int index);
```

```
};
```

For example:

Test	Result
`ArrayList<int> arr;` `for (int i = 0; i < 10; ++i) {` `arr.add(i);` `}` `arr.removeAt(0);` `cout << arr.toString() << '\n';` `cout << arr.size();`	`[1, 2, 3, 4, 5, 6, 7, 8, 9]` `9`
`ArrayList<int> arr;` `for (int i = 0; i < 10; ++i) {` `arr.add(i);` `}` `arr.removeAt(9);` `cout << arr.toString() << '\n';` `cout << arr.size();`	`[0, 1, 2, 3, 4, 5, 6, 7, 8]` `9`

Test	Result
`ArrayList<int> arr;`	`[0, 1, 2, 3, 4, 6, 7, 8, 9]`
`for (int i = 0; i < 10; ++i) {`	`9`
`    arr.add(i);`	
`}`	
`arr.removeAt(5);`	
`cout << arr.toString() << '\n';`	
`cout << arr.size();`	

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

Reset answer

```cpp
template <class T>
T ArrayList<T>::removeAt(int index)
{
    /*
        FUNCTION removeAt(index):
            IF index is less than 0 OR index is greater than or equal to count:
                THROW std::out_of_range("index is out of range")
            END IF

            SET removed_value to data[index]

            FOR i from index to count - 2, incrementing by 1:
                SET data[i] to data[i + 1]
            END FOR

            DECREMENT count by 1

            RETURN removed_value
        END FUNCTION
    */
    if (index < 0 || index >= this->count)
    {
        throw std::out_of_range("the input index is out of range!");
    }
    T removed_data = data[index];
    for (int i = index; i < count-1; i++)
    {
        data[i] = data[i+1];
    }
    count--;
    return removed_data;

}

template <class T>
bool ArrayList<T>::removeItem(T item)
{
    /*
        FUNCTION removeItem(item):
            FOR i from 0 to count - 1, incrementing by 1:
                IF data[i] is equal to item:
                    CALL removeAt(i)
                    RETURN true
                END IF
            END FOR

            RETURN false
        END FUNCTION
    */
   for (int i = 0; i < count; i++)
   {
        if (data[i] == item)
        {
            this->removeAt(i);
            return true;
        }
    }
    return false;
}

template <class T>
void ArrayList<T>::clear()
```

	Test	Expected	Got	

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeAt(0);`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size();`
		`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
		`9`
			`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
			`9`
				✓

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeAt(9);`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size();`
		`[0, 1, 2, 3, 4, 5, 6, 7, 8]`
		`9`
			`[0, 1, 2, 3, 4, 5, 6, 7, 8]`
			`9`
				✓

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeAt(5);`
	
	`cout << arr.toString() << '\n';`
	`cout << arr.size();`
		`[0, 1, 2, 3, 4, 6, 7, 8, 9]`
		`9`
			`[0, 1, 2, 3, 4, 6, 7, 8, 9]`
			`9`
				✓

Passed all tests! ✓

Đúng

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

Câu hỏi **3**

Sai

Không chấm điểm

Implement methods **Get, set, clear, empty, indexOf, contains** in template class **ArrayList** representing the array list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class ArrayList {
protected:
    T* data;        // dynamic array to store the list's items
    int capacity;   // size of the dynamic array
    int count;      // number of items stored in the array
public:
```

```
 ArrayList(){capacity = 5; count = 0; data = new T[5];}
   ~ArrayList(){ delete[] data; }
```

```
    void    add(T e);
    void    add(int index, T e);
    int     size();
    bool    empty();
    void    clear();  //remove data and set the list to the initial condition
    T       get(int index);  //get the element at the index, if the index is out of range, "throw std::out_of_range("index
is out of range");"
```

```
    void    set(int index, T e);  //set the index position in the list with the value e
    int     indexOf(T item);  //get the first index of item in the list, else return -1
    bool    contains(T item);   //check if the item is in the list
    T       removeAt(int index);
    bool    removeItem(T item);
```

```
};
```

Notice: You just have to implement the methods: set, get, clear, empty, indexOf, contains. Other methods have been implemented already.

For example:

<table>
<tr><th>Test</th><th>Result</th></tr>
<tr><td><pre>
    ArrayList<int> arr;
    int size = 10;
    for(int index = 0; index < size; index++){
        arr.add(index);
    }
    cout << arr.toString() << '\n';
    arr.set(0,100);
    cout << arr.get(0) << '\n';
    cout << arr.toString() << '\n';
    arr.clear();
    cout << arr.toString() << '\n';
    cout << arr.empty() << '\n';
    for(int index = 0; index < size; index++){
        arr.add(index);
    }
    cout << arr.indexOf(7) << '\n';
    cout << arr.contains(15) << '\n';
</pre></td><td><pre>
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
100
[100, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[]
1
7
0
</pre></td></tr>
</table>

Test	Result

`ArrayList<int> arr;`
`    int size = 10;`
`    for(int index = 0; index < size; index++){`
`        arr.add(index);`
`    }`
`    try {`
`        arr.set(10,100);`
`        cout << arr.get(10) << '\n';`
`    }`
`    catch(const std::exception & e){`
`        cout << e.what() << endl;`
`    }`
	`Index is out of range`

Answer:

```
template <class T>
void ArrayList<T>::clear()
{
    /*
        FUNCTION clear:
            IF data is not NULL:
                DELETE data
            END IF

            SET count to 0
            SET capacity to 5
            CREATE new array with size capacity
        END FUNCTION
    */
    if (data != nullptr)
    {
        delete[] this->data;
    }
    this->count = 0;
    this->capacity = 5;
    T *new_data = new T[this->capacity];
    this->data = new_data;
}
template <class T>
T ArrayList<T>::get(int index)
{
    /*
        FUNCTION get(index):
            IF index is less than 0 OR index is greater than or equal to count:
                THROW std::out_of_range("index is out of range")
            END IF

            RETURN data[index]
        END FUNCTION
    */
    if (index < 0 || index >= count)
    {
        throw std::out_of_range("index is out of range");
    }
    return data[index];
}

template <class T>
void ArrayList<T>::set(int index, T e)
{
    /*
        FUNCTION set(index, element):
            IF index is less than 0 OR index is greater than or equal to count:
                THROW std::out_of_range("index is out of range")
            END IF

            SET data[index] to element
        END FUNCTION
    */
    if (index < 0 || index >= count)
    {
        throw std::out_of_range("index is out of range");
    }
    data[index] = e;
}

template <class T>
```

	Test	Expected	Got	

✗	`ArrayList<int> arr;`
	`int size = 10;`
	`for(int index = 0; index < size; index++){`
	`arr.add(index);`
	`}`
	`cout << arr.toString() << '\n';`
	`arr.set(0,100);`
	`cout << arr.get(0) << '\n';`
	`cout << arr.toString() << '\n';`
	`arr.clear();`
	`cout << arr.toString() << '\n';`
	`cout << arr.empty() << '\n';`
	`for(int index = 0; index < size; index++){`
	`arr.add(index);`
	`}`
	`cout << arr.indexOf(7) << '\n';`
	`cout << arr.contains(15) << '\n';`
		[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
		100
		[100, 1, 2, 3, 4, 5, 6, 7, 8, 9]
		[]
		1
		7
		0
			[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
			100
			[100, 1, 2, 3, 4, 5, 6, 7, 8, 9]
			[]
			0
			7
			0
				✗

✗	`ArrayList<int> arr;`
	`int size = 10;`
	`for(int index = 0; index < size; index++){`
	`arr.add(index);`
	`}`
	`try {`
	`arr.set(10,100);`
	`cout << arr.get(10) << '\n';`
	`}`
	`catch(const std::exception & e){`
	`cout << e.what() << endl;`
	`}`
		Index is out of range	index is out of range	✗

Testing was aborted due to error.

Show differences

Câu hỏi **4**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of integers `nums` and a two-dimension array of integers `operations`.
Each operation in `operations` is represented in the form `{L, R, X}`. When applying an operation, all elements with index in range `[L, R]` (include `L` and `R`) increase by `X`.
Your task is to implement a function with following prototype:

```
vector<int> updateArrayPerRange(vector<int>& nums, vector<vector<int>>& operations);
```

The function returns the array after applying all operation in `operations`.

Note:
- The `iostream`, and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<int> nums {13, 0, 6, 9, 14, 16};`
`vector<vector<int>> operations {{5, 5, 16}, {3, 4, 0}, {0, 2, 8}};`
`printVector(updateArrayPerRange(nums, operations));`
	`[21, 8, 14, 9, 14, 32]`

Answer: (penalty regime: 0 %)

Reset answer

```

// Implement function
vector<int> updateArrayPerRange(vector<int> &nums, vector<vector<int>> &operations)
{
    // FUNCTION updateArrayPerRange(nums, operations):
    //     FOR each operation in operations:
    //         SET L to operation[0]
    //         SET R to operation[1]
    //         SET X to operation[2]

    //         FOR i from L to R:
    //             INCREMENT nums[i] by X
    //         END FOR
    //     END FOR

    //     RETURN nums
    // END FUNCTION
    for (vector<int> &operation : operations)
    {
        int L = operation[0];
        int R = operation[1];
        int X = operation[2];

        for (int i = L; i <= R; i++)
        {
            nums[i] += X;
        }
    }

    return nums;
}

```

	Test	Expected	Got	
✓	`vector<int> nums {13, 0, 6, 9, 14, 16};` `vector<vector<int>> operations {{5, 5, 16}, {3, 4, 0}, {0, 2, 8}};` `printVector(updateArrayPerRange(nums, operations));`	`[21, 8, 14, 9, 14, 32]`	`[21, 8, 14, 9, 14, 32]`	✓
✓	`vector<int> nums {19, 4, 3, 2, 16, 3, 17, 8, 18, 12};` `vector<vector<int>> operations {{0, 3, 4}, {2, 5, 12}, {3, 6, 6}, {5, 8, 5}, {8, 9, 8}, {0, 5, 9}, {1, 7, 8}, {1, 1, 3}, {5, 5, 18}};` `printVector(updateArrayPerRange(nums, operations));`	`[32, 28, 36, 41, 51, 61, 36, 21, 31, 20]`	`[32, 28, 36, 41, 51, 61, 36, 21, 31, 20]`	✓

Passed all tests! ✓

Đúng

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

Câu hỏi **5**

Đúng một phần

Đạt điểm 0,60 trên 1,00

Given an array of integers.
Your task is to implement a function with the following prototype:

```
bool consecutiveOnes(vector<int>& nums);
```

The function returns if all the `1`s appear consecutively in `nums`. If `nums` does not contain any elements, please return `true`

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` are being used. No other libraries are allowed.
- You can write helper functions.
- Do not use global variables in your code.

For example:

Test	Result

`vector<int> nums {0, 1, 1, 1, 9, 8};`
`cout << consecutiveOnes(nums);`
	1

Answer: (penalty regime: 0 %)

Reset answer

```
bool consecutiveOnes(vector<int>& nums){
    if (nums.empty())
    {
        return true;
    }
    bool IsSequence = false;
    for (int i = 0; i < nums.size(); i++)
    {
        if (nums[i] == 1 & nums[i] == nums[i+1])
        {
            return true;
        }

    }
    return false;
}
```

	Test	Expected	Got	

✓	`vector<int> nums {0, 1, 1, 1, 9, 8};`
	`cout << consecutiveOnes(nums);`
		1	1	✓

✓	`vector<int> nums {};`
	`cout << consecutiveOnes(nums);`
		1	1	✓

Your code failed one or more hidden tests.

Đúng một phần

Marks for this submission: 0,60/1,00.

Câu hỏi **6**

Đúng một phần

Đạt điểm 0,90 trên 1,00

The prices of all cars of a car shop have been saved as an array called N. Each element of the array N is the price of each car in shop. A person, with the amount of money k want to buy as much cars as possible.

Request: Implement function

buyCar(int* nums, int length, int k);

Where `nums` is the array N, `length` is the size of this array and `k` is the amount of money the person has. Find the maximum cars this person can buy with his money, and return that number.

Example:

`nums=[90, 30, 20, 40, 50]; k=90;`

The result is 3, he can buy the cars having index 1, 2, 3 (first index is 0).

Note: The library `iostream`, `'algorithm'` and `using namespace std` have been used. You can add other functions but you are not allowed to add other libraries.

For example:

Test	Result

`int nums[] = {90,30,40,90,20};`
`int length = sizeof(nums)/sizeof(nums[0]);`
`cout << buyCar(nums, length, 90) << "\n";`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int buyCar(int* nums, int length, int k){
    for (int i = length - 1; i > 0; i--)
    {
        for (int j = i - 1; j >= 0; j--)
        { // Corrected the loop condition
            if (nums[i] < nums[j])
            {
                int temp = nums[j];
                nums[j] = nums[i];
                nums[i] = temp;
            }
        }
    }
    int carBought = 0;
    int totalSpent = k;
    for (int i = 0; i < length - 1; i++)
    {
        if (totalSpent >= nums[i])
        {
            totalSpent -= nums[i];
            carBought++;
        }
        else{
            break;
        }
    }
    return carBought;
}
```

	Test	Expected	Got	

✓	`int nums[] = {90,30,40,90,20};`
	`int length = sizeof(nums)/sizeof(nums[0]);`
	`cout << buyCar(nums, length, 90) << "\n";`
		3	3	✓

Your code failed one or more hidden tests.

Đúng một phần

Marks for this submission: 0,90/1,00.

Câu hỏi **7**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of integers.
Your task is to implement a function with following prototype:

```
int equalSumIndex(vector<int>& nums);
```

The function returns the smallest index `i` such that the sum of the numbers to the left of `i` is equal to the sum of the numbers to the right.
If no such index exists, return `-1`.

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<int> nums {3, 5, 2, 7, 6, 4};`
`cout << equalSumIndex(nums);`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int equalSumIndex(vector<int>& nums) {
    int totalSum = 0;
    int leftSum = 0;

    // Calculate the total sum of the array
    for (int num : nums) {
        totalSum += num;
    }

    // Iterate through the array to find the equal sum index
    for (int i = 0; i < nums.size(); i++) {
        // Calculate rightSum by subtracting leftSum and the current element from totalSum
        int rightSum = totalSum - leftSum - nums[i];

        if (rightSum == leftSum) {
            return i;
        }

        // Update leftSum for the next iteration
        leftSum += nums[i];
    }

    return -1; // Return -1 if no such index is found
}
```

	Test	Expected	Got	

✓	`vector<int> nums {3, 5, 2, 7, 6, 4};`
	`cout << equalSumIndex(nums);`
		3	3	✓

✓	`vector<int> nums {3};`
	`cout << equalSumIndex(nums);`
		0	0	✓

Passed all tests! ✓

Đúng

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

Câu hỏi **8**

Đúng

Đạt điểm 1,00 trên 1,00

Given an array of strings.
Your task is to implement a function with following prototype:

```
int longestSublist(vector<string>& words);
```

The function returns the length of the longest subarray where all words share the same first letter.

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result

`vector<string> words {"faction", "fight", "and", "are", "attitude"};`
`cout << longestSublist(words);`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int longestSublist(vector<string>& words) {
    if (words.empty()) {
        return 0;
    }

    int longest = 0;
    int currentLength = 1;
    char currentChar = words[0][0];

    for (int i = 1; i < words.size(); ++i) {
        if (words[i][0] == currentChar) {
            ++currentLength;
        } else {
            if (currentLength > longest) {
                longest = currentLength;
            }
            currentLength = 1;
            currentChar = words[i][0];
        }
    }

    if (currentLength > longest) {
        longest = currentLength;
    }

    return longest;
}
```

	Test	Expected	Got	

✓	`vector<string> words {"faction", "fight", "and", "are", "attitude"};`
	`cout << longestSublist(words);`
		3	3	✓

✓	`vector<string> words {};`
	`cout << longestSublist(words);`
		0	0	✓

Passed all tests! ✓

Đúng

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