Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Ba, 11 tháng 2 2025, 2:47 PM
Kết thúc lúc	Chủ Nhật, 23 tháng 2 2025, 11:46 PM
Thời gian thực hiện	12 Các ngày 8 giờ
Điểm	7,00/7,00
Điểm	**10,00** trên 10,00 (**100**%)

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

```
template <class T>
void ArrayList<T>::ensureCapacity(int cap) {
    if (cap > capacity) {
        int newCapacity = static_cast<int>(capacity * 1.5);
        if (newCapacity < cap) {
            newCapacity = cap;
        }
        T* newData = new T[newCapacity];
        for (int i = 0; i < count; i++) {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        capacity = newCapacity;
    }
}
```

```
18
19  template <class T>
20  void ArrayList<T>::add(T e) {
21      ensureCapacity(count + 1);
22      data[count++] = e;
23  }
24
25  template <class T>
26  void ArrayList<T>::add(int index, T e) {
27      if (index < 0 || index > count) {
28          throw std::out_of_range("the input index is out of range!");
29      }
30      ensureCapacity(count + 1);
31      for (int i = count; i > index; i--) {
32          data[i] = data[i - 1];
33      }
34      data[index] = e;
35      count++;
36  }
37
38  template <class T>
39  int ArrayList<T>::size() {
40      return count;
41  }
42
```

	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`
				✓

✓	`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;
 	
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

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

Reset answer

```
template <class T>
T ArrayList<T>::removeAt(int index) {
    if (index < 0 || index >= count) {
        throw std::out_of_range("index is out of range");
    }

    T removedValue = data[index];  // Store the element to return it later

    // Shift elements to the left
    for (int i = index; i < count - 1; i++) {
        data[i] = data[i + 1];
    }

    count--;  // Reduce the size of the list
    return removedValue;
}

template <class T>
bool ArrayList<T>::removeItem(T item) {
    for (int i = 0; i < count; i++) {
        if (data[i] == item) {  // Found the first occurrence
            removeAt(i);  // Reuse removeAt to delete it
            return true;
        }
    }
    return false;  // Item not found
}

template <class T>
void ArrayList<T>::clear() {
    delete[] data;  // Free the old memory
    capacity = 5;   // Reset capacity
    count = 0;      // Reset count
    data = new T[capacity];  // Allocate a new empty array
}


```

	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**

Không trả lời

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:

Test	Result

`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`

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:

```
1
```

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

```
vector<int> updateArrayPerRange(vector<int>& nums, vector<vector<int>>& operations) {
    int n = nums.size();
    vector<int> diff(n + 1, 0); // Difference array

    // Apply range updates using the difference array
    for (const auto& op : operations) {
        int L = op[0], R = op[1], X = op[2];
        diff[L] += X;
        if (R + 1 < n) {
            diff[R + 1] -= X;
        }
    }

    // Apply the difference array to the original array
    int increment = 0;
    for (int i = 0; i < n; i++) {
        increment += diff[i];
        nums[i] += increment;
    }

    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

Đạt điểm 1,00 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) {
    int first = -1, last = -1;

    // Find the first and last occurrence of 1
    for (int i = 0; i < nums.size(); i++) {
        if (nums[i] == 1) {
            if (first == -1) first = i;  // First occurrence of 1
            last = i;                    // Last occurrence of 1 (keeps updating)
        }
    }

    // If there are no 1s, return true
    if (first == -1) return true;

    // Check if all elements between first and last 1 are also 1s
    for (int i = first; i <= last; i++) {
        if (nums[i] != 1) return false;
    }

    return true;
}

```

	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	✓

Passed all tests! ✓

Đúng

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

Câu hỏi **6**

Đúng

Đạt điểm 1,00 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) {
    // Sort the prices in ascending order
    std::sort(nums, nums + length);

    int count = 0;  // Number of cars bought
    for (int i = 0; i < length; i++) {
        if (k >= nums[i]) {
            k -= nums[i];  // Deduct the price from budget
            count++;        // Increment count of purchased cars
        } else {
            break;  // Stop if we can't afford the next car
        }
    }

    return count;
}
```

	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	✓

Passed all tests! ✓

Đúng

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

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, leftSum = 0;

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

    // Iterate through the array to find the equilibrium index
    for (int i = 0; i < nums.size(); i++) {
        int rightSum = totalSum - leftSum - nums[i];  // Compute right sum
        if (leftSum == rightSum) return i;  // Check equilibrium condition
        leftSum += nums[i];  // Update left sum for next iteration
    }

    return -1;  // No equilibrium index 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 maxLen = 1, currentLen = 1;
    char prevChar = words[0][0];  // First letter of first word

    for (int i = 1; i < words.size(); i++) {
        if (words[i][0] == prevChar) {
            currentLen++;  // Continue the current sequence
        } else {
            maxLen = max(maxLen, currentLen);  // Update max length
            currentLen = 1;  // Reset sequence length
            prevChar = words[i][0];  // Update the first letter
        }
    }

    return max(maxLen, currentLen);  // Final update
}

```

	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.