Đã bắt đầu vào lúc	Thứ bảy, 30 Tháng chín 2023, 3:13 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Chủ nhật, 1 Tháng mười 2023, 6:09 PM
Thời gian thực hiện	1 ngày 2 giờ
Điểm	7,00/7,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 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(nums, nums + length);
  int count = 0;
  int i = 0;
  while (i < length && k >= nums[i]) {
    k -= nums[i];
    count++;
    i++;
  }

  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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Chính xác

Điểm 1,00 của 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;
    }
    int head = 0, tail = 0;
    for (int i = 0; i <= nums.size(); ++i)
    {
        if (nums[i] == 1)
        {
            head = i;
            break;
        }
    }
    for (int i = 0; i <= nums.size(); ++i)
    {
        if (nums[i] == 1)
            tail = i;
    }
    bool consecutive = true;
    for (int i = head; i <= tail && head != 0; ++i)
```

	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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 của 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 left_sum = 0;
    int right_sum = 0;
    for (int i = 0; i < nums.size(); i++)
    {
        right_sum += nums[i];
    }
    for (int i = 0; i < nums.size(); i++)
    {
        right_sum -= nums[i];
        if (left_sum == right_sum)
        {
            return i;
        }
        left_sum += nums[i];
    }
    return -1;
}
```

	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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

Chính xác

Điểm 1,00 của 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)
{
    int n = words.size();
    int c1 = 0, c2 = 0;
    for (int i = 0; i < n; i++)
    {
        c1 = 0;
        for (int j = i; j < n; j++)
        {
            if (words[i][0] == words[j][0] || words[i][0] == words[j][0] - 32 || words[i][0] == wo
                c1++;
            else
                break;
        }
        if (i == 0)
            c2 = c1;
        else if (c1 > c2)
            c2 = c1;
    }
    return c2;
}
```

	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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 5

Chính xác

Điểm 1,00 của 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:
            new_capacity = capacity * 1.5;
            create new array with new_capacity
        else: do nothing
    */
    if (cap >= capacity)
    {
        int newCapacity = capacity * 1.5;
        T *newData = new T[newCapacity];
        for (int i = 0; i < count; i++)
        {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
        capacity = newCapacity;
    }
}

template <class T>
void ArrayList<T>::add(T e)
{
    /* Insert an element into the end of the array. */
    ensureCapacity(count + 1);
    data[count++] = e;
}

template <class T>
void ArrayList<T>::add(int index, T e)
{
    /*
        Insert an element into the array at given index.
        if index is invalid:
            throw std::out_of_range("the input index is out of range!");
    */
    if (index < 0 || index > count)
    {
        throw std::out_of_range("the input index is out of range!");
    }

    ensureCapacity(count + 1);
    for (int i = count; i > index; i--)
    {
        data[i] = data[i - 1];
    }
    data[index] = e;
    count++;
}

```

```
template <class T>
int ArrayList<T>::size()
{
    /* Return the length (size) of the array */
    return 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`
				✓

✓	`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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 6

Chính xác

Điểm 1,00 của 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
	```

```
    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)
{
    /*
    Remove element at index and return removed value
    if index is invalid:
        throw std::out_of_range("index is out of range");
    */
    if (index < 0 || index >= count)
    {
        throw std::out_of_range("the input index is out of range!");
    }

    T removedItem = data[index];
    for (int i = index; i < count - 1; i++)
    {
        data[i] = data[i + 1];
    }
    count--;
    return removedItem;
}

template <class T>
bool ArrayList<T>::removeItem(T item)
{
    /* Remove the first apperance of item in array and return true, otherwise return false */
    for (int i = 0; i < count; i++)
    {
        if (data[i] == item)
        {
            removeAt(i);
            return true;
        }
    }
    return false;
}

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

```
{
    /*
        Delete array if array is not NULL
        Create new array with: size = 0, capacity = 5
    */
    if (data != nullptr)
    {
        delete[] data;
    }

    capacity = 5;
    count = 0;
    data = new T[5];
}
```

	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
				✓

	Test	Expected	Got	

✓	`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
				✓

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

	Test	Expected	Got	

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

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeItem(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.removeItem(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
				✓

	Test	Expected	Got	

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeItem(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`
				✓

✓	`ArrayList<int> arr;`
	
	`for (int i = 0; i < 10; ++i) {`
	`    arr.add(i);`
	`}`
	`arr.removeItem(-5);`
	
	`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 = 10;
	for(int idx=0; idx < size; idx++){
	    arr.add(idx);
	}
	int values[] = {10, 15, 2, 6, 4, 7, 40, 8};
	// 0 1 2 3 4 5 6 7
	int index[] = {0, 1, 5, 3, 2, 1, 1, 0};
	
	/* 10, 15, 2, 6, 4, 7, 40, 8 //initial list
	    * 15, 2, 6, 4, 7, 40, 8 //after removeAt 0
	    * 15, 6, 4, 7, 40, 8 //after removeAt 1
	    * 15, 6, 4, 7, 40 //after removeAt 5
	    * 15, 6, 4, 40 //after removeAt 3
	    * 15, 6, 40 //after removeAt 2
	    * 15, 40 //after removeAt 1
	    * 15, //after removeAt 1
	    * {} //after removeAt 0
	    */
	
	arr.clear();
	for(int idx=0; idx < 8; idx++)
	    arr.add(values[idx]);
	
	//removeAt:
	for(int idx=0; idx < 8; idx++){
	    int idxRemoved = index[idx];
	    arr.removeAt(idxRemoved);
	    //check expected values
	}
		[]
		0
			[]
			0
				✓

	Test	Expected	Got	

	`cout << arr.toString() << '\n';`
	`cout << arr.size();`			

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 1,00 của 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 sizeop = operations.size();
    for (int i = 0; i < sizeop; i++)
    {
        for (int index = operations[i][0]; index <= operations[i][1]; index++)
        {
            nums[index] += operations[i][2];
        }
    }
    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! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.