Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Bảy, 13 tháng 4 2024, 3:38 PM
Kết thúc lúc	Thứ Ba, 16 tháng 4 2024, 1:50 PM
Thời gian thực hiện	2 Các ngày 22 giờ

Câu hỏi **1**
Đúng
Đạt điểm 1,00

Implement function

```
int binarySearch(int arr[], int left, int right, int x)
```

to search for value x in array arr using recursion.
After traverse an index in array, we print out this index using cout << "We traverse on index: " << index << endl;

Note that middle of left and right is floor((right-left)/2)

For example:

Test	**Result**

`int arr[] = {1,2,3,4,5,6,7,8,9,10};`
`int x = 10;`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int result = binarySearch(arr, 0, n - 1, x);`
`(result == -1) ? cout << "Element is not present in array"`
`                : cout << "Element is present at index " << result;`
	`We traverse on index: 4`
	`We traverse on index: 7`
	`We traverse on index: 8`
	`We traverse on index: 9`
	`Element is present at index 9`

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

Reset answer

```
int binarySearch(int arr[], int left, int right, int x)
{
    if (left > right) return -1;
    int middle = left + (right - left)/2;
    if (arr[middle] == x) {
        cout << "We traverse on index: " << middle << endl;
        return middle;
    } else if (arr[middle] > x) {
        cout << "We traverse on index: " << middle << endl;
        return binarySearch(arr, left, middle - 1, x);
    } else {
        cout << "We traverse on index: " << middle << endl;
        return binarySearch(arr, middle + 1, right, x);
    }
}
```

	Test	Expected	Got	

✓	`int arr[] = {1,2,3,4,5,6,7,8,9,10};`
	`int x = 10;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int result = binarySearch(arr, 0, n - 1, x);`
	`(result == -1) ? cout << "Element is not present in array"`
	`                : cout << "Element is present at index "`
	`<< result;`
		`We traverse on index: 4`
		`We traverse on index: 7`
		`We traverse on index: 8`
		`We traverse on index: 9`
		`Element is present at`
		`index 9`
			`We traverse on index: 4`
			`We traverse on index: 7`
			`We traverse on index: 8`
			`We traverse on index: 9`
			`Element is present at`
			`index 9`
				✓

Passed all tests! ✓

Câu hỏi **2**
Đúng
Đạt điểm 1,00

Implement function

```
int interpolationSearch(int arr[], int left, int right, int x)
```

to search for value x in array arr using recursion.
After traverse to an index in array, before returning the index or passing it as argument to recursive function, we print out this index using cout << "We traverse on index: " << index << endl;

Please note that you can't using key work for, while, goto (even in variable names, comment).

For example:

Test	Result

`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int x = 3;`
`int result = interpolationSearch(arr, 0, n - 1, x);`
`(result == -1) ? cout << "Element is not present in array"`
`               : cout << "Element is present at index " << result;`
	`We traverse on index: 2`
	`Element is present at index 2`

`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int x = 0;`
`int result = interpolationSearch(arr, 0, n - 1, x);`
`(result == -1) ? cout << "Element is not present in array"`
`               : cout << "Element is present at index " << result;`
	`Element is not present in array`

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

Reset answer

```
int interpolationSearch(int arr[], int left, int right, int x)
{
    int pos;
    if (left <= right && arr[left] <= x && arr[right] >= x){
        pos = left + (((double)(right-left)/(arr[right]-arr[left]))*(x-arr[left]));
        if (arr[pos] == x){
        cout << "We traverse on index: " << pos << endl;
        return pos;
        }
        if (arr[pos] < x){
        cout << "We traverse on index: " << pos << endl;
        return interpolationSearch(arr, pos + 1, right, x);
        }
        if (arr[pos] > x){
        cout << "We traverse on index: " << pos << endl;
        return interpolationSearch(arr, left, pos - 1, x);
    }
    return -1;
}
```

	Test	Expected	Got	

✓	`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int x = 3;`
	`int result = interpolationSearch(arr, 0, n - 1, x);`
	`(result == -1) ? cout << "Element is not present in array"`
	`               : cout << "Element is present at index " << result;`
		We traverse on index: 2
		Element is present at index 2
			We traverse on index: 2
			Element is present at index 2
				✓

✓	`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int x = 0;`
	`int result = interpolationSearch(arr, 0, n - 1, x);`
	`(result == -1) ? cout << "Element is not present in array"`
	`               : cout << "Element is present at index " << result;`
		Element is not present in array	Element is not present in array	✓

Passed all tests! ✓

Câu hỏi **3**
Đúng
Đạt điểm 1,00

In computer science, a jump search or block search refers to a search algorithm for ordered lists. The basic idea is to check fewer elements (than linear search) by jumping ahead by fixed steps or skipping some elements in place of searching all elements. For example, suppose we have an array arr[] of size n and block (to be jumped) size m. Then we search at the indexes arr[0], arr[m], arr[2m].....arr[km] and so on. Once we find the interval (arr[km] < x < arr[(k+1)m]), we perform a linear search operation from the index km to find the element x. The optimal value of m is √n, where n is the length of the list.

In this question, we need to implement function jumpSearch with step √n to search for value x in array arr. After searching at an index, we should print that index until we find the index of value x in array or until we determine that the value is not in the array.

```
int jumpSearch(int arr[], int x, int n)
```

For example:

Test	Result

`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 };`
`int x = 55;`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int index = jumpSearch(arr, x, n);`
 	
`if (index != -1) {`
`    cout << "\nNumber " << x << " is at index " << index;`
`}`
`else {`
`    cout << "\n" << x << " is not in array!";`
`}`
	`0 4 8 12 9 10`
	`Number 55 is at index 10`

`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 };`
`int x = 144;`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int index = jumpSearch(arr, x, n);`
 	
`if (index != -1) {`
`    cout << "\nNumber " << x << " is at index " << index;`
`}`
`else {`
`    cout << "\n" << x << " is not in array!";`
`}`
	`0 4 8 12`
	`Number 144 is at index 12`

Answer: (penalty regime: 0 %)

Reset answer

```
int jumpSearch(int arr[], int x, int n) {
    int i = 0, jump = sqrt(n);
    for (i = 0; i < n; i += jump) {
        cout << i << " ";
        if (arr[i] > x)
            break;
        else if (arr[i] == x)
            return i;
    }
    for (int j = i - jump + 1; j < i; j++) {
        cout << j << " ";
        if (arr[j] == x)
            return j;
    }
    return -1;
}
```

	Test	Expected	Got	

✓	`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 };`
	`int x = 55;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int index = jumpSearch(arr, x, n);`
	
	`if (index != -1) {`
	`    cout << "\nNumber " << x << " is at index " << index;`
	`}`
	`else {`
	`    cout << "\n" << x << " is not in array!";`
	`}`
		0 4 8 12 9 10
		Number 55 is at index 10
			0 4 8 12 9 10
			Number 55 is at index 10
				✓

	Test	Expected	Got	

✓	`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 };`
	`int x = 144;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int index = jumpSearch(arr, x, n);`
	
	`if (index != -1) {`
	`    cout << "\nNumber " << x << " is at index " << index;`
	`}`
	`else {`
	`    cout << "\n" << x << " is not in array!";`
	`}`
		`0 4 8 12`
		`Number 144 is at index 12`
			`0 4 8 12`
			`Number 144 is at index 12`
				✓

✓	`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 611, 612, 613 };`
	`int x = 612;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int index = jumpSearch(arr, x, n);`
	
	`if (index != -1) {`
	`    cout << "\nNumber " << x << " is at index " << index;`
	`}`
	`else {`
	`    cout << "\n" << x << " is not in array!";`
	`}`
		`0 4 8 12 16 17`
		`Number 612 is at index 17`
			`0 4 8 12 16 17`
			`Number 612 is at index 17`
				✓

✓	`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 611, 612, 613 };`
	`int x = 614;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int index = jumpSearch(arr, x, n);`
	
	`if (index != -1) {`
	`    cout << "\nNumber " << x << " is at index " << index;`
	`}`
	`else {`
	`    cout << "\n" << x << " is not in array!";`
	`}`
		`0 4 8 12 16 17 18 19`
		`614 is not in array!`
			`0 4 8 12 16 17 18 19`
			`614 is not in array!`
				✓

	Test	Expected	Got	

✓	`int arr[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 611, 612, 613, 1000, 1002, 2000, 2003, 2004, 2005, 2006 };`
	`int x = 36;`
	`int n = sizeof(arr) / sizeof(arr[0]);`
	`int index = jumpSearch(arr, x, n);`
	
	`if (index != -1) {`
	`    cout << "\nNumber " << x << " is at index " << index;`
	`}`
	`else {`
	`    cout << "\n" << x << " is not in array!";`
	`}`
		0 5 10 6 7 8 9
		36 is not in array!
			0 5 10 6 7 8 9
			36 is not in array!
				✓

Passed all tests! ✓

Câu hỏi **4**
Đúng
Đạt điểm 1,00

Given an array of distinct integers, find if there are two pairs (a, b) and (c, d) such that a+b = c+d, and a, b, c and d are distinct elements. If there are multiple answers, you can find any of them.

Some libraries you can use in this question:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <iostream>
#include <utility>
#include <map>
#include <vector>
#include <set>
```

Note: The function checkAnswer is used to determine whether your pairs found is true or not in case there are two pairs satistify the condition. You don't need to do anything about this function.

For example:

Test	Result

`int arr[] = { 3, 4, 7, 1, 2, 9, 8 };`
`int n = sizeof arr / sizeof arr[0];`
`pair<int, int> pair1, pair2;`
`if (findPairs(arr, n, pair1, pair2)) {`
`    if (checkAnswer(arr, n, pair1, pair2)) {`
`        printf("Your answer is correct.\n");`
`    }`
`    else printf("Your answer is incorrect.\n");`
`}`
`else printf("No pair found.\n");`
	`Your answer is correct.`

Test	Result

`int arr[] = { 3, 4, 7 };`
`int n = sizeof arr / sizeof arr[0];`
`pair<int, int> pair1, pair2;`
`if (findPairs(arr, n, pair1, pair2)) {`
`    if (checkAnswer(arr, n, pair1, pair2)) {`
`        printf("Your answer is correct.\n");`
`    }`
`    else printf("Your answer is incorrect.\n");`
`}`
`else printf("No pair found.\n");`
	`No pair found.`

Answer: (penalty regime: 0 %)

Reset answer

```
bool findPairs(int arr[], int n, pair<int,int>& pair1, pair<int, int>& pair2)
{
    map<int, pair<int, int>> m;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int sum = arr[i] + arr[j];
            if (m.find(sum) == m.end())
                m[sum] = make_pair(i, j);
            else {
               pair<int, int> p = m[sum];
               pair1 = {arr[p.first], arr[p.second]};
               pair2 = {arr[i], arr[j]};
               return true;
            }
        }
    }
    return false;
}
```

	Test	Expected	Got	

✓	`int arr[] = { 3, 4, 7, 1, 2, 9, 8 };`
	`int n = sizeof arr / sizeof arr[0];`
	`pair<int, int> pair1, pair2;`
	`if (findPairs(arr, n, pair1, pair2)) {`
	`    if (checkAnswer(arr, n, pair1, pair2)) {`
	`        printf("Your answer is correct.\n");`
	`    }`
	`    else printf("Your answer is incorrect.\n");`
	`}`
	`else printf("No pair found.\n");`
		`Your answer is correct.`	`Your answer is correct.`	✓

✓	`int arr[] = { 3, 4, 7 };`
	`int n = sizeof arr / sizeof arr[0];`
	`pair<int, int> pair1, pair2;`
	`if (findPairs(arr, n, pair1, pair2)) {`
	`    if (checkAnswer(arr, n, pair1, pair2)) {`
	`        printf("Your answer is correct.\n");`
	`    }`
	`    else printf("Your answer is incorrect.\n");`
	`}`
	`else printf("No pair found.\n");`
		`No pair found.`	`No pair found.`	✓

Passed all tests! ✓