Không hoàn thành

Chấm điểm của 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};	We traverse on index: 4
int x = 10;	We traverse on index: 7
<pre>int n = sizeof(arr) / sizeof(arr[0]);</pre>	We traverse on index: 8
<pre>int result = binarySearch(arr, 0, n - 1, x);</pre>	We traverse on index: 9
<pre>(result == -1) ? cout << "Element is not present in array"</pre>	Element is present at index 9
<pre>: cout << "Element is present at index " << result;</pre>	

Answer: (penalty regime: 0 %)

Reset answer

Precheck

Không hoàn thành

Chấm điểm của 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:

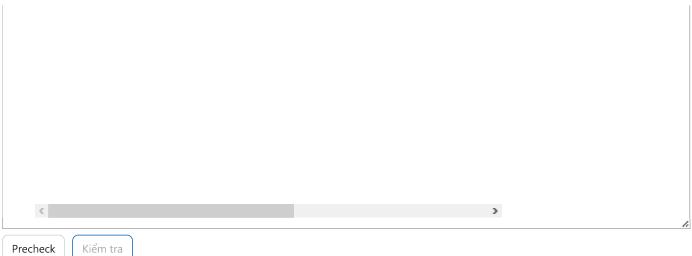
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <iostream>
#include <utility>
#include <map>
#include <vector>
#include <set>
```

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
<pre>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"); }</int,></pre>	Your answer is correct.
<pre>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");</int,></pre>	No pair found.

Answer: (penalty regime: 0 %)



Không hoàn thành

Chấm điểm của 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 %)

```
int foldShift(long long key, int addressSize)

v {
    int rotation(long long key, int addressSize)

v {
    int rotation(long long key, int addressSize)
}
```

Không hoàn thành

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

Answer: (penalty regime: 0 %)

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

v
{
3
4
}
```

Không hoàn thành

Chấm điểm của 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 \sqrt{n} , where n is the length of the list.

In this question, we need to implement function jumpSearch with step \sqrt{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.

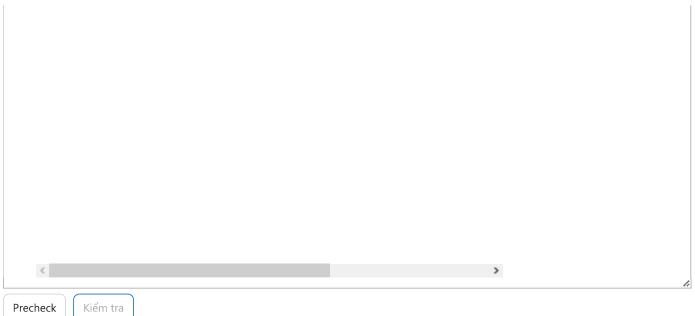
```
int jumpSearch(int arr[], int x, int n)
```

For example:

Test	Result
<pre>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);</pre>	0 4 8 12 9 10 Number 55 is at index 10
<pre>if (index != -1) { cout << "\nNumber " << x << " is at index " << index; } else { cout << "\n" << x << " is not in array!"; }</pre>	
<pre>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);</pre>	0 4 8 12 Number 144 is at index 12
<pre>if (index != -1) { cout << "\nNumber " << x << " is at index " << index; } else { cout << "\n" << x << " is not in array!"; }</pre>	

Answer: (penalty regime: 0 %)

```
1 int jumpSearch(int arr[], int x, int n) {
2    // TODO: print the traversed indexes and return the index of value x in {
3  }
```



Không hoàn thành

Chấm điểm của 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
<pre>int a[]={1,2,5}; cout << digitExtraction(122443,a,3);</pre>	223
cout < <midsquare(9452);< td=""><td>3403</td></midsquare(9452);<>	3403

Answer: (penalty regime: 0 %)

**

Precheck

Không hoàn thành

Chấm điểm của 1,00

There are n people, each person has a number between 1 and 100000 (1 \le n \le 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:

BÁCH KHOA E-LEARNING



WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

♀ 268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Reset answer

