Problemas a serem resolvidos como prova final:

1. The Interpolation Search is an improvement over Binary Search for instances, where the values in a sorted array are uniformly distributed. Binary Search always goes to middle element to check. On the other hand interpolation search may go to different locations according the value of key being searched. For example if the value of key is closer to the last element, interpolation search is likely to start search toward the end side.
   To find the position to be searched, it uses following formula.

```
// The idea of formula is to return higher value of pos
// when element to be searched is closer to arr[hi]. And
// smaller value when closer to arr[lo]
pos = lo + [ (x-arr[lo])*(hi-lo) / (arr[hi]-arr[Lo]) ]

arr[] ==> Array where elements need to be searched
x     ==> Element to be searched
lo    ==> Starting index in arr[]
hi    ==> Ending index in arr[]
```

Algorithm
Rest of the Interpolation algorithm is same except the above partition logic.
Step1: In a loop, calculate the value of “pos” using the probe position formula.
Step2: If it is a match, return the index of the item, and exit.
Step3: If the item is less than arr[pos], calculate the probe position of the left sub-array. Otherwise calculate the same in the right sub-array.
Step4: Repeat until a match is found or the sub-array reduces to zero.

2. Given a sorted array and a number x, find a pair in array whose sum is closest to x. Examples:

```
Input: arr[] = {10, 22, 28, 29, 30, 40}, x = 54

Output: 22 and 30

Input: arr[] = {1, 3, 4, 7, 10}, x = 15

Output: 4 and 10
```

Following is detailed algorithm.

```
1) Initialize a variable diff as infinite (Diff is used to store the
   difference between pair and x).  We need to find the minimum diff.
2) Initialize two index variables l and r in the given sorted array.
     (a) Initialize first to the leftmost index:  l = 0
     (b) Initialize second  the rightmost index:  r = n-1
3) Loop while l < r.
     (a) If  abs(arr[l] + arr[r] - sum) < diff  then
         update diff and result
     (b) Else if(arr[l] + arr[r] <  sum )  then l++
     (c) Else r--
```

3. Encontrar floor e ceil de um número x dentro de um array a. O número x pode não estar no array a. O floor(x) é o número do array a que é menor que x e que mais se aproxima de x (pode existir mais de um número menor que x, o floor é o maior deles). Dualmente, o ceil(x) é o número do array a que é maior que x e que mais se aproxima de x (pode existir maisde um número maior do que x, o ceil é o menor deles).
4. The name of this searching algorithm may be misleading as it works in O(Log n) time. The name comes from the way it searches an element.

```
Given a sorted array an element x to be
searched, find position of x in the array.


Input:  arr[] = {10, 20, 40, 45, 55}
       x = 45
Output: Element found at index 3
Input:  arr[] = {10, 15, 25, 45, 55}
       x = 15
Output: Element found at index 1
```

Exponential search involves two steps:

a. Find range where element is present
b. Do Binary Search in above found range.

How to find the range where element may be present?
The idea is to start with subarray size 1 compare its last element with x, then try size 2, then 4 and so on until last element of a subarray is not greater.
Once we find an index i (after repeated doubling of i), we know that the element must be present between i/2 and i (Why i/2? because we could not find a greater value in previous iteration)

5. Implementar a estrutura de dados Heap e seus algoritmos (usando datatype) em Haskell
6. Implementar uma pilha e seus algoritmos em Haskell
7. Implementar uma Fila e seus algoritmos em Haskell
8. Given an array and a number k where k is smaller than size of array, we need to find the k'th smallest element in the given array. It is given that ll array elements are distinct. Examples:

```
Input: arr[] = {7, 10, 4, 3, 20, 15}
       k = 3
Output: 7


Input: arr[] = {7, 10, 4, 3, 20, 15}
       k = 4
Output: 10
```

9. Inversion Count for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.

Two elements a[i] and a[j] form an inversion if

a[i] > a[j] and i < j. For simplicity, we may

assume that all elements are unique.

Example:

Input: arr[] = {8, 4, 2, 1}

Output: 6

Given array has six inversions (8,4), (4,2),

(8,2), (8,1), (4,1), (2,1).

10. O conjunto das partes de um conjunto S, nomeado como $\wp(S)$ é definido como um conjunto contendo todos os subconjuntos de S. Por exemplo, $S = \{a, b\}$, então $\wp(S) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$. Implemente um algoritmo que recebe um conjunto de elementos guardados em um array e retorna seu conjunto das partes.