Prob 2:
a)

```
struct stack {
   unsigend short size (2)
   T minimum (sizeof T)
   T data[maxSize] (sizeof T * maxSize)
  }
```

Following type T, the struct stack need size that sizeof (T * (max size + 1) + 4) bytes. Size of stack always be n >= 0. It couldn't be negative. If max size is bigger then 65535, size could be int or more bigger type.

b)
In initializing, size = 0 is required.

PUSH(T) -> add T at index 'size' of array data of stack. if size is 0, minimum is T, and size + 1.
else T is smaller then minimum in stack, minimum is T. Or not ignore.
POP() -> return T at index 'size' of array data of stack. And size -1. If size is 0, return error type of T.
TOP() -> return T at index 0 of array data of stack. If size is 0, return error type of T.
SIZE() -> return size of stack.
isEmpty() -> return size of stack. In programming, 0 means false, else means true.
getMinimum() -> return minimum of stack.

c) all of function needs O(1) Time, follow a), need size that sizeof (T * (max size + 1) + 4) bytes.

Prob 3:
a) Trevarse from last index to first index. On each data(height of lighthouse) traverse current index of data to index 0, start with data - 1 and if data[k] is bigger then data[i], data - 1, else escape inner traverse and so on.

b)

```
int [] func(int arr[]) {
   for (int k = arr.size() - 1; k >=0; k—) {
      if ( k == 0 ) {
         arr[k] = -1;
         break;
      }

      for( int i =  k-1; i >= 0; i—) {
         arr[k]—;

         if (arr[k] > arr[i]) arr[k]—;
         else break;
      }
   }
   return arr;
}
```

c) O(log n)