COMP3512 – Lab Exercise 5 (Oct 13–16, 2015)

Name: ______________________________ Set ______

1. Implement a function `kth_selection`

```
// precondition: v.size() > 0 && 1 <= k && k <= v.size()
int kth_selection(const vector<int>& v, size_t k);
```

that returns the kth smallest number in the vector v without sorting the vector or calling any algorithm in the STL.

Note that if k is 1, the function returns the minimum of v.

The idea of the algorithm is as follows: "Divide" v into 3 parts (vectors): `left`, `mid` & `right`. `left` contains those numbers in v less than `v[0]`, `mid` contains those equal to `v[0]` & `right` contains those greater than `v[0]`. Then depending on the value of k, the kth smallest number of v is either in `left`, `mid` or `right`. Now use recursion.

2. We would like to generate all subsets of a set of n elements. Each subset can be represented by a string of n "bits", 0's or 1's, where 0 indicates that the corresponding element is not in the subset & 1 indicates that it is. For example, the string 0111 (for a set of 4 elements) indicates that the first element is not in the subset, but the second, third & fourth elements are, i.e., it represents the subset containing only the second, third & fourth elements.

For a set of 4 elements, there 16 possible subsets represented by 16 "bit strings". It is possible to list all 16 strings in such a way that only 1 bit changes between successive strings. For example,

0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000

Notice that only 1 bit changes between successive values of the sequence.

In the general case, let S_n be the sequence of n-bit strings where only 1 bit changes each time. Then S_n can be recursively defined by the following 2 rules:

$$\begin{aligned} S_0 &= \epsilon \\ S_{n+1} &= 0S_n, 1S_n^R \end{aligned}$$

In the above, ϵ denotes the empty string, $0S_n$ denotes the sequence with 0 prefixed to each string of S_n & $1S_n^R$ denotes the sequence S_n in reverse order with 1 prefixed to each string. Note that S_n consists of 2^n strings.

Write a function

```
vector<string> subsets(size_t n);
```

that implements the above recursive algorithm to generate the sequence S_n. The function returns a vector containing the strings in the sequence.

ecked: ______________________________