Sets (e.g. $S = \{2, 4, 6, 8, 10\}$.)

Main applications involve answering questions of the form "is $x \in S$?".

How to represent S in code?

What if $S \subseteq U$ ($U \equiv$ "universe")
and $|U|$ is "small"?

$|S| \equiv$ # elements of S

Idea 1: just store all elements in a vector.
works, but not easy to answer
"is $x \in S$" - might take $\approx |S|$ steps.

Idea 2: same, but sort the vector & use binary search. Better, but hard to modify S.

Another way to look at sets $S \subseteq U$:
Since sets can't have duplicate elements,
each set defines a unique function

$\chi_S : U \longrightarrow \{0,1\}$ which completely characterizes S.

(χ_S called the "characteristic" function.)

$$\left\{ \begin{array}{c} \text{functions from} \\ U \to \{0,1\} \end{array} \right\} \longleftrightarrow \left\{ S \subseteq U \right\}$$

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{else.} \end{cases}$$

How to get S from χ_S?

$$S = \chi_S^{-1}(\{1\})$$

How to represent functions from $U \to \{0,1\}$ if $|U|$ is small? Say U = set of all characters ($|U| = 256$).

Could represent as an array or vector<bool>

```
vector<bool> XS;
// idea: XS[x] == true
//       <=>  x ∈ S.
// First make S empty:
for (int i = 0; i < 256; i++)
        XS.push_back(false);

// Now XS represents the empty set.
// add an element
XS[x] = true;  // Now x ∈ S
// (note: we assume x of type char)
```

```
char x;
cin >> x;
if (XS[x]) cout << x << " in S";
else cout << x << " not in S";
```

```
// XS['a'] = XS['e'] = ... = XS['u'] = true;
//Now XS stores vowels.
```

Answering "$x \in S$?" very efficient!

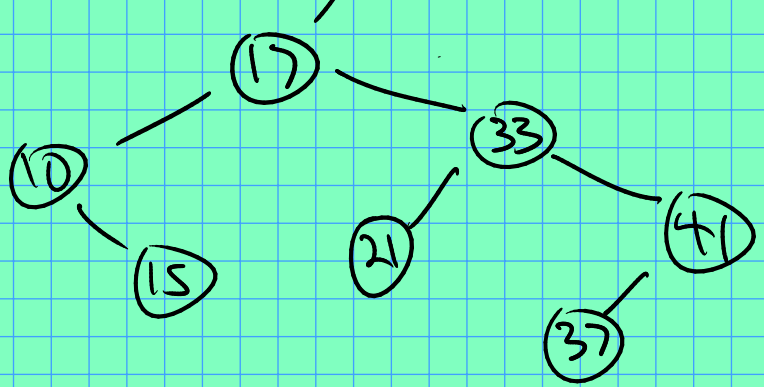
Downside: required (potentially) lots of memory (Need vector/array of $|U|$ elements).

STL sets: similar to Idea 2 above (sorted vector), but adding/removing elements is efficient ($\approx \log_2 |S|$ steps)

Idea: use a tree:

if $S = \{10, 20, 15, 17, 41, 33, 37, 21\}$

behind the scenes you might have a tree like this



(Picture for the array representation:)