1 Tables Concepts

2 Lookup Tables
   2.1. Set of Integers as a Lookup Table ........................................ 3
   2.2. Set of Strings as a Lookup Table .......................................... 5
   2.3. Strengths and Weaknesses of Lookup Tables ............................ 7

3 Hash Tables
1. Tables Concepts

- Tables have rows and columns
- Rows are usually identified by a name (number or even a string)
- Columns are usually identified by a name (number or even a string)
- Rows may have multiple columns of data
- Tables are indexed to access row data at a certain column

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table as a Set ADT

Table as a Map ADT
2. Lookup Tables

- A lookup table is a table in which all potential data is known
- A lookup table is accessed by simple array indexing in O(1) time
- Lookup tables may be multidimensional

2.1. Set of Integers as a Lookup Table

- A set of unsigned integers in the range [0, 4]
- Implemented internally as a simple array of booleans

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interface

```cpp
class LookupTableInt {
    public:
        LookupTableInt();
        void add ( unsigned int v );
        void remove ( unsigned int v );
        bool contains ( unsigned int v );

    private:
        static const unsigned int size = 5;
        bool m_table[size];
};
```
### Implementation

```cpp
LookupTableInt::LookupTableInt()
{
}

void LookupTableInt::add( unsigned int v )
{
    m_table[v] = true;
}

void LookupTableInt::remove( unsigned int v )
{
    m_table[v] = false;
}

bool LookupTableInt::contains( unsigned int v )
{
    return m_table[v];
}
```

### Main

```cpp
int main( void )
{
    LookupTableInt table;

    table.add(3);   // Add value 3
    table.remove(3); // Remove value 3

    return 0;
}
```
2.2. Set of Strings as a Lookup Table

- A set of strings limited to "Apple", "Banana", "Cherry"
- Implemented internally as a simple array of booleans
- Must correspond each possible string to an internal array index

<table>
<thead>
<tr>
<th>Index</th>
<th>Apple</th>
<th>Banana</th>
<th>Cherry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interface

```cpp
class LookupTableString
{
    public:
        LookupTableString();

    void add ( std::string v );
    void remove ( std::string v );
    bool contains ( std::string v );

    private:
        unsigned int ind( std::string v );
        static const unsigned int size = 3;
        bool m_table[size];
};
```

Implementation

```cpp
LookupTableString::LookupTableString()
{
    
}
2. Lookup Tables

2.2. Set of Strings as a Lookup Table

```cpp
void LookupTableString::add( std::string v )
{
    m_table[ ind(v) ] = true;
}

void LookupTableString::remove( std::string v )
{
    m_table[ ind(v) ] = false;
}

bool LookupTableString::contains( std::string v )
{
    return m_table[ ind(v) ];
}

unsigned int LookupTableString::ind( std::string v )
{
    if ( v == "Apple" ) return 0;
    else if ( v == "Banana" ) return 1;
    else if ( v == "Cherry" ) return 2;
    else throw std::out_of_range( "No such fruit!" );
}

Main

int main( void )
{
    LookupTableString table;

    table.add( "Cherry" ); // Add value "Cherry"
    table.remove( "Cherry" ); // Remove value "Cherry"

    return 0;
}
```
2.3. **Strengths and Weaknesses of Lookup Tables**

- We have implemented a set of integers as a lookup table
- We have implemented a set of strings as a lookup table
- Recall that lists, vectors, and even trees can also implement sets!

<table>
<thead>
<tr>
<th>Set ADT as a List</th>
<th>Set ADT as a Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>add</strong></td>
<td><strong>remove</strong></td>
</tr>
<tr>
<td>list/vector</td>
<td>O(N)</td>
</tr>
<tr>
<td>tree</td>
<td>O(log(N))</td>
</tr>
<tr>
<td>lut</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

**Strengths of Lookup Tables**

- Lookup tables have great time complexity!

**Weaknesses of Lookup Tables**

- I want a set for a larger range of numbers (e.g., from 0 to 1,000,000)
  - Lookup tables are ______________
- I want a set of other types (e.g., strings)
  - Lookup tables are ______________