CS 211
Instructor: Kangmei Yang
<table>
<thead>
<tr>
<th>Decimals (Base 10)</th>
<th>Binary (Base 2)</th>
<th>Hexadecimal (Base 16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>10001</td>
<td>11</td>
</tr>
</tbody>
</table>
Negative numbers

The most-significant bit is a sign bit to represent positive and negative numbers.

Example if we only have 4 bits,

- 0000 $\leftarrow$ represents 0
- 0001 $\leftarrow$ represents 1

If we just change the sign bit, then 1000 should represent negative 0?
Well, there is no positive or negative 0, we prefer 0000 to represent 0.
Thus negate the bit, then plus 1.

- 0000

<table>
<thead>
<tr>
<th>Negate bits</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plus 1</td>
<td>+ 1</td>
</tr>
<tr>
<td></td>
<td>10000 $\leftarrow$ Since we only have 4 bits Leading 1 is neglected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Negative 1 $\rightarrow$ 0001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negate bits 1110</td>
</tr>
<tr>
<td>Plus 1 + 1</td>
</tr>
<tr>
<td>1111</td>
</tr>
</tbody>
</table>
Towers of Hanoi

For every move of ring n, ring n-1 has to move \(2^{n-1}\) times.

If we have 5 rings,
Ring 5 move once
Ring 4 move 2 times
Ring 3 move 4 times
Ring 2 move 8 times
Ring 1 move 16 times
Thus, in total, \(1+2+4+8+16\)
= \(2^0+2^1+2^2+2^3+2^4\)
In binary 11111
We know that if we plus 1, we’ll get 100000
Which is \(2^5\)
So for 5 rings, there are \(2^5 - 1\) moves
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> t[3];  //array of 3 vectors that stores int
    // three towers A,B,C represented as an array of 3 vectors
    int n, candidate, to, from, move=0;
    // move counts the move number
    cout<<"Please enter an ODD number of rings to move: ";
    cin>>n;

    /*We have one virtue large ring at the bottom of each tower for
     the ease of comparison.
     fill the tower with rings,
     tower A has all the disc, t[0].push_back(1) function add 1 to the
     back of tower A*/
    for(int i = n+1;i >=1;i--)
        t[0].push_back(i);
    t[1].push_back(n+1);
    t[2].push_back(n+1);
Towers of Hanoi
Odd Rings

/* initialize towers and candidate
   for odd rings, the first move is to move the top ring from the
   source tower to the destination tower */

from=0;
to=1;
candidate=1;

while( t[1].size() ) {
    /* As long as the destination tower doesn’t have all the rings
       size() is an instance function that returns how many rings in the
       vector*/
Towers of Hanoi
Odd Rings

/* print out the instruction for current step
we already setup the towers from and to, also the candidate ring to move
A has ascii value 65
B has ascii value 66
C has ascii value 67
Since t[0] is tower A, t[1] is tower B, t[2] is tower C,
We’d like to print A for index 0, B for index 1, C for index 2
*/
cout <<"move number "+move<<": Transfer ring 
<<candidate<< " from tower "<< char(from+65)<<" to tower 
"<<char(to+65)<<endl;

//mimic the move in vectors
1. Push the ring to the “to” tower

2. Remove the ring from the “from” tower
t[from].pop_back();
Towers of Hanoi
Odd Rings

// Setup for the next move,
// look for the ring to move
// we’re not going to move the ring that we just moved
// we’re about to move the smaller ring of the other two towers
// other towers →

///(to + 1) % 3 index of the right tower
///(to + 2) % 3 index of the left tower
// check the last(top) ring of each tower
if (t[(to + 1) % 3].back() < t[(to + 2) % 3].back())
  from =;
else
  from =;
// set the index to the tower with the smaller ring
candidate = t[from].back(); // get the ring to move from the tower
if (candidate % 2 == 1) // odd ring move left
to = ;
else // even ring move right
to = ;
// This part of move will be different for even n