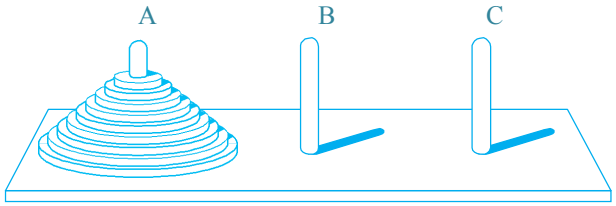
Examples of Non-numeric Recursion

The Towers of Hanoi and Dry Bones examples that follow demonstrate that recursion is not limited to functions that return numerical values. Instead, recursion can be applied to solve any of the wide variety of problems whose solutions are inherently recursive.

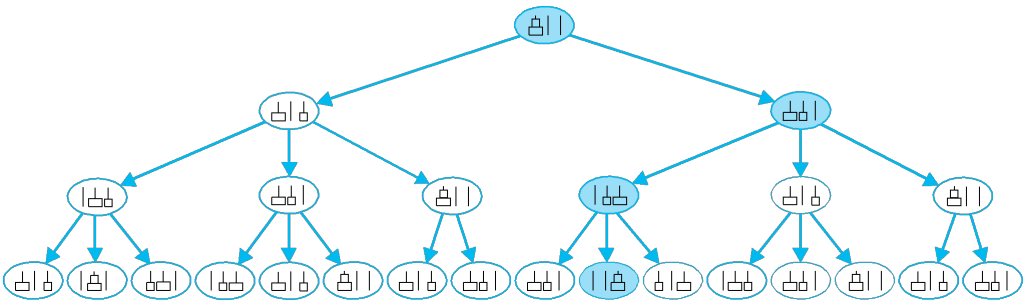
Example: Towers of Hanoi

The **Towers of Hanoi** problem is to solve the puzzle shown in the following figure, in which one must move the disks from the left peg to the right peg according to the following rules:

1. When a disk is moved, it must be placed on one of the three pegs.
2. Only one disk may be moved at a time, and it must be the top disk on one of the pegs.
3. A larger disk may never be placed on top of a smaller one.



The following *game tree* shows the various configurations that are possible in the problem with two disks; the highlighted path in the tree shows a solution to the two-disk problem:



Legend has it that the priests in the Temple of Bramah were given a puzzle consisting of a golden platform with three diamond needles on which were placed sixty-four golden disks. The priests were to move one disk per day, following the preceding rules, and when they had successfully finished moving the disks to another needle, time would end. (*Question:* If the priests moved one disk per day and began their work in year 0, when would time end?)

Novices usually find the puzzle easy to solve for a small number of disks, but they have more difficulty as the number of disks grows to seven, eight, and beyond. To a computer scientist, however, the Towers of Hanoi puzzle is easy: We begin by identifying a base case, for which the problem is trivial to solve:

If there is one disk, then move it from Peg A to Peg C.

The puzzle is thus easily solved for disk. We then seek an inductive solution for $n > 1$ disks, in which we assume that a solution exists for $n - 1$ disks:

1. *Move the topmost* $n - 1$ *disks from Peg A to Peg B, using Peg C for temporary storage.*
2. *Move the final disk remaining on Peg A to Peg C.*
3. *Move the* $n - 1$ *disks from Peg B to Peg C, using Peg A for temporary storage.*

This scheme is implemented by the recursive function `move()` in the following program that solves the Towers of Hanoi puzzle recursively.

```
/* hanoitowers.cpp solves the Towers of Hanoi puzzle recursively.

   Input:  numDisks, the number of disks to be moved
   Output: a sequence of moves that solve the puzzle
-----------------------------------------------------------------*/

#include <iostream>
using namespace std;

void move(int n, char source, char destination, char spare);

int main()
{
  const char PEG1 = 'A',              // the three pegs
             PEG2 = 'B',
             PEG3 = 'C';

  cout << "This program solves the Hanoi Towers puzzle.\n\n";

  cout << "Enter the number of disks: ";
  int numDisks;                      // the number of disks to be moved
  cin >> numDisks;
  cout << endl;

  move(numDisks, PEG1, PEG2, PEG3);  // the solution
}

/* move is a recursive function to solve the Hanoi Towers puzzle.

    Receive:
      n, the number of disks to be moved;
      source, the needle the disks are to be moved from,
      destination, the needle the disks are to be moved to, and
      spare, the needle that can be used to store disks temporarily
-----------------------------------------------------------------*/

void move(int n, char source, char destination, char spare)
{
  if (n <= 1)                                  // anchor
    cout << "Move the top disk from " << source << " to "
         << destination << endl;
  else
  {                                           // inductive case
    move(n-1, source, spare, destination);
    move(1, source, destination, spare);
    move(n-1, spare, destination, source);
  }
}
```

Sample Runs:

```
This program solves the Hanoi Towers puzzle.

Enter the number of disks: 3

Move the top disk from A to B
Move the top disk from A to C
Move the top disk from B to C
Move the top disk from A to B
Move the top disk from C to A
Move the top disk from C to B
Move the top disk from A to B


This program solves the Hanoi Towers puzzle.

Enter the number of disks: 4

Move the top disk from A to C
Move the top disk from A to B
Move the top disk from C to B
Move the top disk from A to C
Move the top disk from B to A
Move the top disk from B to C
Move the top disk from A to C
Move the top disk from A to B
Move the top disk from C to B
Move the top disk from C to A
Move the top disk from B to A
Move the top disk from C to B
Move the top disk from A to C
Move the top disk from A to B
Move the top disk from C to B
```

Example: Dry Bones!

The Old Testament book of Ezekiel is a book of vivid images that chronicle the siege of Jerusalem by the Babylonians and the subsequent forced relocation (known as the *exile*) of the Israelites following Jerusalem's fall. Chapter 37 describes a powerful vision of Ezekiel in which a valley of dry bones becomes reconnected as he prophesies. Tendons, muscle, and skin regrow on the reconnected bones and the dead skeletons return to life forming a vast army that fills the valley. This vision of new life arising from dry bones provided the homesick Israelites with hope that the" dry bones" of defeated Israel would "come back to life" and they could one day be free of their oppressors, the Babylonians.

Before the American Civil War, many black men and women in the United States found themselves in a situation similar to that of the Israelites, having been forcibly relocated from their homelands and brought to the United States to serve as slaves. One way that they were able to keep their hopes alive was through singing songs that spoke of freedom. However, songs that expressed such hopes directly could result in a beating for the singer, and so these sentiments had to be cleverly encoded in lyrics that would not catch the attention of the slaves' overseers.

One way this was done was by singing **spirituals**—songs whose lyrics referred to stories from the Bible that carried themes of hope and freedom. Songs like *Go Down Moses; Swing Low,*

Sweet Chariot; Wade in the Water; and *Bright Canaan* are examples of spirituals that, in addition to their overt spiritual message, carried a second level of meaning that provided these men and women with the hope that they could one day be free of oppression.

One of the most subtle spirituals is *Dry Bones*. It makes no direct references to freedom, but uses the imagery of scattered bones being reconnected to encode the same message of hope conveyed in Ezekiel 37. With its message of freedom carefully hidden, such a song could be sung in the presence of even the harshest overseer:

```
Ezekiel cried, "Dem dry bones!"
Ezekiel cried, "Dem dry bones!"
Ezekiel cried, "Dem dry bones!"
Oh, hear the word of the Lord.
The foot bone connected to the leg bone,
The leg bone connected to the knee bone,
The knee bone connected to the thigh bone,
The thigh bone connected to the back bone,
The back bone connected to the neck bone,
The neck bone connected to the head bone,
Oh, hear the word of the Lord!
Dem bones, dem bones gonna walk aroun'
Dem bones, dem bones, gonna walk aroun'
Dem bones, dem bones, gonna walk aroun'
Oh, hear the word of the Lord.
The head bone connected to the neck bone,
The neck bone connected to the back bone,
The back bone connected to the thigh bone,
The thigh bone connected to the knee bone,
The knee bone connected to the leg bone,
The leg bone connected to the foot bone,
Oh, hear the word of the Lord!
Dem bones, dem bones gonna walk aroun'
Dem bones, dem bones, gonna walk aroun'
Dem bones, dem bones, gonna walk aroun'
Oh, hear the word of the Lord.
```

The structure of the song is interesting, because it can be partitioned into the following steps, which together comprise an algorithm for printing the song:

a. Print the `"Ezekiel cried"` variation of the chorus.

b. Print the *bone lyrics* from foot to head.

c. Print the `"Dem bones"` variation of the chorus.

d. Print the *bone lyrics* from head to foot.

e. Print the `"Dem bones"` variation of the chorus.

Because of the reversal in steps b and d, the actions performed in steps b-d can be described using recursion with step c as an anchor case.

```
/* dryBones.cpp displays the lyrics of the song "Dry Bones."

  Output: lyrics of "Dry Bones"
-----------------------------------------------------------------*/

#include <iostream>                       // cin, cout, <<, >>
#include <string>                         // string class
using namespace std;

void printChorus(const string& variation);
void printLastLine();
void printBoneLyrics(const string & bone);

int main()
{
  printChorus("Ezekiel cried, \"Dem dry bones!\"\n");
  printBoneLyrics("foot");
  printLastLine();
  printChorus("Dem bones, dem bones gonna walk aroun'\n");
}


/* printChorus displays the lyrics of the chorus of the song, of
   which there are two variations.

   Receive: variation, a string
   Output:  the chorus with the specified variation.
-----------------------------------------------------------------*/

void printChorus(const string & variation)
{
  cout << variation << variation << variation;
  printLastLine();
}


/* printLastLine displays the last line of each verse and the chorus.

   Output:  the last line
-----------------------------------------------------------------*/

void printLastLine()
{
  cout << "Oh, hear the word of the Lord!\n\n";
}


/* printBoneLyrics displays the lyrics for a given bone.

   Receive: bone, a string.
   Output:  the lyrics for that bone and (recursively) the lyrics
            for the bones "beneath" it (during winding) and then
            for the bones "above" it (during unwinding).
-----------------------------------------------------------------*/
```

```
string getNext(const string& aBone);

void printBoneLyrics(const string& bone)
{
  if (bone == "head")                    // Anchor: bone == head
                                         //  sing chorus variation
    printChorus("Dem bones, dem bones gonna walk aroun'\n");

  else
  {                                      // Ind-Step: bone < head
    string nextBone = getNext(bone);     //  find next body part

    cout << "The " << bone               //  do 'upward' lyric
         << " bone connected to the "    //  before recursion
         << nextBone << " bone,\n";      //  (winding)

    printBoneLyrics(nextBone);           //  do rest recursively

    cout << "The " << nextBone           //  do 'downward' lyric
         << " bone connected to the "    //  after recursion
         << bone << " bone,\n";          //  (unwinding)
  }
}


/* getNext() gets the next bone.

   Receive:      aBone, a string.
   Precondition: aBone is a valid bone (in the song).
   Return:       the bone above aBone.
-----------------------------------------------------------------*/

string getNext(const string& aBone)
{
  if (aBone == "foot")
    return "leg";
  else if (aBone == "leg")
    return "knee";
  else if (aBone == "knee")
    return "thigh";
  else if (aBone == "thigh")
    return "back";
  else if (aBone == "back")
    return "neck";
  else if (aBone == "neck")
    return "head";
  else
    cerr << "\n*** GetNext(): "
         << aBone << " is unknown!" << endl;
  return "";
}
```