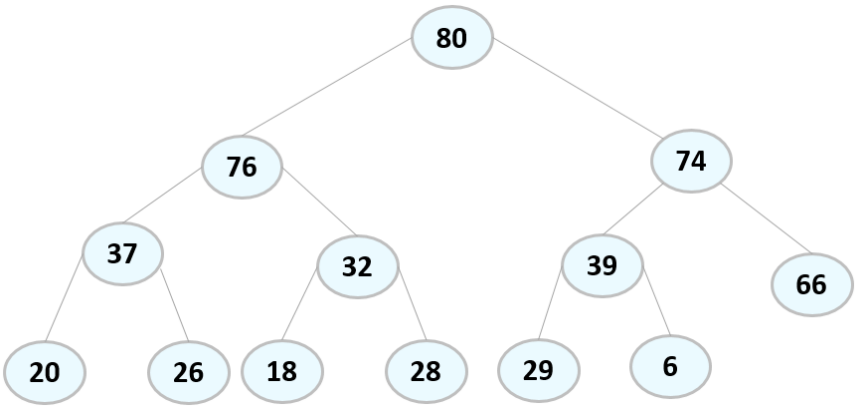
## Assignment 6

1. (Weight: 20%) Show the result of removing the number 80 from the following heap. Show the new heap and its array representation.




2. (Weight: 10%) The largest object should be removed first from a priority queue. Explain why operator less is used as the default to compare objects being placed in a priority queue instead of operator greater.

3. (Weight: 20%) **Programming:** Write a Compare function class that inserts Person objects in a priority queue based on the number of dependents a person has. The Person object with the largest number of dependents should be removed first.

**Hint:** In order to use the priority queue class, we need to pass in a structure that allows comparing the objects. If the max queue keeps track of primitive values (e.g. integers, strings), then the built-in C++ less than will compare the objects fine. However, since we want to compare persons, we need to tell the queue how to compare them. Now, the compare function class is simply a structure that overrides the parenthesis operator, and the operator takes two persons as parameters. Then, the persons are compared based on the number of dependents. Since this is a max queue, we should use the < operator.

Here's an example of writing a compare structure (from Huffman Tree): (however, this one is for a min priority queue)

```
struct Compare_Huffman_Trees
{
bool operator()(const Binary_Tree<Huff_Data>& left_tree, constBinary_Tree<Huf
f_Data>& right_tree) {

double wLeft = left_tree.get_data().weight;
double wRight = right_tree.get_data().weight;
return wLeft > wRight;
}
};
```

4. (Weight: 20%) Create the Huffman code tree for the following frequency table:

| Symbol | Frequency |
|---|---|
| * | 50 |
| + | 30 |
| - | 25 |
| / | 10 |
| % | 5 |

5. (Weight: 10%) What would the Huffman code look like if all symbols in the alphabet had equal frequency?
6. (Weight: 20%) **Programming:** Use the generic Huffman_Tree<T> on Blackboard to encode the words in your favorite nursery rhyme.