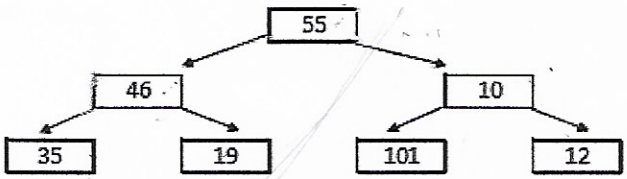
CSE 330 Winter 2020 Midterm – Take-Home Portion

February 12, 20202

Instructor: Kerstin Voigt

Name: Colin Morris-Moncada1 Student ID: 0062796592

The C++ code below is a (almost complete) implementation of "breadth-first traversal" of a binary tree structure. The Queue data structure makes it very easy to code this type of stepping through the nodes of a graph. The binary tree that has been "hard-coded" into the program (see below) looks like this:



The int main() shown below is to traverse this graph "breadth-first" and prints out the value of each visited node. The output for this example will be: 55 46 10 35 19 101 12

The Algorithm: Initialize a queue with the root node. Then while the queue is not empty, print the value of the front node in the queue, remove that node from the queue and add to the back of the queue the nodes that are to the left (if there is) and right (if there is) of the node whose value was just printed. (For some nodes there are no nodes to their left or right, so nothing gets added to thequeue.)

Fill in the missing pieces of code to complete this implementation of breadth-first tree traversal.

```
template <typename T>
class NODE
{
public:

    NODE()
        :left(nullptr), right(nullptr) {}

    NODE(T x)
        :value(x), left(nullptr), right(nullptr) {}

    NODE(T x, NODE<T>* lft, NODE<T>* rgt)
        : value(x), left(lft), right(rgt) {}

    T value;
    NODE<T>* left;
    NODE<T>* right;
};

int main()
{
    NODE<int> n35(35);
    NODE<int> n19(19);
    NODE<int> n101(101);
    NODE<int> n12(12);
```

1. While queue is not empty
   1. print value of front node
   2. Remove that NODE from the queue
   3. Add nodes that are to left and right to back of queue