Suppose that you are searching for the key 70 in a binary search tree. In the following list, circle the letters corresponding to sequences that could not be the sequence of keys examined.

A. 70
B. 99 10 80 20 60 70
C. 10 80 40 32 50 70
D. 10 26 30 48 50 62 70
E. 41 99 20 85 70
F. 44 80 55 61 70
G. 99 11 53 86 70
H. 22 58 81 70

Suppose that we create a BST by inserting the letters A through G into an initially empty tree. In the blank to the left of each of the following insertion orders write the height of the tree produced when keys are inserted in that order into an initially empty tree. Height is defined as the max distance from the root to any node. (e.g., A tree that has only its root and one child has height 2.) The first answer is provided for you.

A. ____ A, B, C, D, E, F, G
B. _____ B, A, C, D, G, F, E
C. _____ D, C, E, B, A, F, G
D. _____ D, B, F, A, C, E, G
E. _____ E, D, C, A, B, G, F
F. _____ C, B, D, A, E, G, F

Consider the following class

```
struct Node{
   Node* next;
};

class Forest
{
   private:
      Node** links;
   public:
      Forest(int n){
         links = new Node*[n];
         for(int i=0; i<n; i++){
            links[i] = new Node;
         }
      }
      Node* root(int i){
         Node* temp = links[i];
         while(temp->next != nullptr)
            temp = temp->next;
         return temp;
      }
      void merge(int i, int j){
         root(i)->next = root(j);
      }
      bool merged(int i, int j){
         return root(i) == root(j);
      }
      ~Forest(){
         delete[] links;
      }
};
```

a. Draw the forest that is created by the following code in the main method:

```
int main(){
Forest t(8);
t.merge(0, 3);
t.merge(1, 2);
t.merge(1, 4);
t.merge(5, 6);
t.merge(3, 4);
t.merge(7, 5);
```

b. Now suppose that the following calls to t.merged() follow the client code of part A. In the blank to the left of each call, write true or false to give the value returned.

__ true__ t.merged(0, 3);
__false__ t.merged(0, 7);
__ true__ t.merged(1, 3);
__false__ t.merged(4, 5);