TUTORIAL 7

STACKS AND QUEUE

1. Describe the two basic operations on a stack.
2. Consider the following statements:

```
stackType<int> stack;
int x, y;
```

Show what is output by the following segment of code:

```
x = 4;
y = 0;
stack.push(7);
stack.push(x);
stack.push(x + 5);
y = stack.top();
stack.pop();
stack.push(x + y);
stack.push(y - 2);
stack.push(3);
x = stack.top();
stack.pop();
cout << "x = " << x << endl;
cout << "y = " << y << endl;
while (!stack.isEmptyStack())
{
    cout << stack.top() << endl;
    stack.pop();
}
```

3. Consider the following statements:

```
stackType<int> stack;
int x;
```

Suppose that the input is:

14 45 34 23 10 5 -999

Show what is output by the following segment of code:

```
stack.push(5);
cin >> x;
while (x != -999)
{
    if (x % 2 == 0)
    {
        if (!stack.isFullStack())
            stack.push(x);
    }

    else
        cout << "x = " << x << endl;
    cin >> x;
}
cout << "Stack Elements: ";
while (!stack.isEmptyStack())
{
    cout << " " << stack.top();
    stack.pop();
}
cout << endl;
```

4. What is the output of the following program?

```
#include <iostream>
#include <string>
#include "myStack.h"
using namespace std;
template <class type>
 void mystery(stackType<type>& s, stackType<type>& t);
 int main()
{
     stackType<string> s1;
     stackType<string> s2;
      string list[] = {"Winter", "Spring", "Summer", "Fall",
                       "Cold", "Warm", "Hot"};

for (int i = 0; i < 7; i++)
   s1.push(list[i]);
   mystery(s1, s2);
while (!s2.isEmptyStack())
{
  cout << s2.top() << " ";
 s2.pop();
}
   cout << endl;
}
template <class type>
void mystery(stackType<type>& s, stackType<type>& t)
{
  while (!s.isEmptyStack())
 {
  t.push(s.top());
  s.pop();
 }
}
```

5. What is the output of the following program?

```
#include <iostream>
#include <string>
#include "myStack.h"
using namespace std;
void mystery(stackType<int>& s, stackType<int>& t);
int main()
{
 int list[] = {5, 10, 15, 20, 25};
stackType<int> s1;
stackType<int> s2;
for (int i = 0; i < 5; i++)
s1.push(list[i]);
mystery(s1, s2);
while (!s2.isEmptyStack())
{
  cout << s2.top() << " ";
 s2.pop();
}
   cout << endl;
}
 void mystery(stackType<int>& s, stackType<int>& t)
{
  while (!s.isEmptyStack())
 {
  t.push(2 * s.top());
```

```
        s.pop();
    }
}
```

6. Explain why, in the linked implementation of a stack, it is not necessary to implement the operation to determine whether the stack is full.

BINARY TREE

7. The first node in a binary tree is called the _____.
8. A binary tree node's left and right pointers point to the node's _____.
9. A node with no children is called a(n) _____.
10. A(n) _____ is an entire branch of the tree, from one particular node down.
11. The three common types of traversal with a binary tree are _____, _____, and _____.
12. Write a pseudocode algorithm for the preorder traversal.
13. Write a pseudocode algorithm for the postorder traversal.
14. Draw a diagram of the resulting binary tree. Suppose the following values are inserted into a binary tree, in the order given:
    12, 7, 9, 10, 22, 24, 30, 18, 3, 14, 20
15. How would the values in the tree you sketched for Question 17 be displayed in an inorder traversal?
16. How would the values in the tree you sketched for Question 17 be displayed in a preorder traversal?
17. How would the values in the tree you sketched for Question 17 be displayed in a postorder traversal?