Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

ICOM 4035 – Data Structures

Fall 2001

Practice Exercises for Final Exam

1. Trace the execution of the following operations on a Stack ("Push y Pop").

```
Stack<string> S;
S.push("Tom");
S.push("Bob");
S.push("Tim");
cout << S.top() << endl;
S.push(S.top());
S.pop();
S.pop();
cout << S.size() << endl;
S.pop();
S.push("Jil");
S.make_empty();
S.push("Tim");
cout << S.top() << endl;
```

2. Trace the execution of the following operation on a Queue.

```
Queue<string> Q;
Q.enqueue("Mike");
Q.enqueue("Jim");
Q.enqueue("Bob");
Q.enqueue("Pim");
Q.enqueue("Rob");
Q.dequeue();
cout << Q.size() << endl;
cout << Q.front() << endl;
Q.enqueue(Q.front());
Q.dequeue();
Q.dequeue();
```

```
if (Q.front() == “Pim”){
        cout << “Pim is next” << endl;
}
else {
        cout << “Pim is not next” << endl;
}
Q.enqueue(“Tere”);
Q.dequeue();
```

3. Trace the execution of the following operations on a Binary Search Tree if integers.

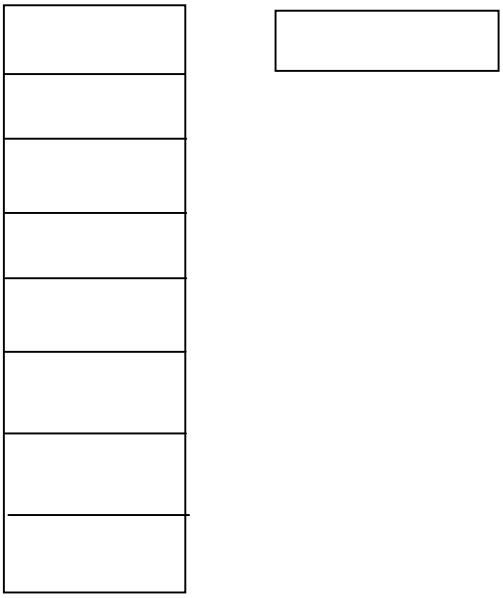
```
BinarySearchTree<int> T;
T.insert(4);
T.insert(4);
T.insert(5);
T.insert(4);
T.insert(-1);
T.insert(3);
T.insert(2);
T.print_in_order();
T.erase(4);
T.find(4);  // return an iterator showing what nodes the queue will have
T.erase(3);
T.insert(3);
T.find(3); // return an iterator showing what nodes the queue will have
cout << T.size() << endl;
T.print_post_order();
T.make_empty();
T.insert(9);
T.insert(10);
```

4. Suppose we have an integer hash table H with 7 bucket, and a hash function *hash()* defined as:

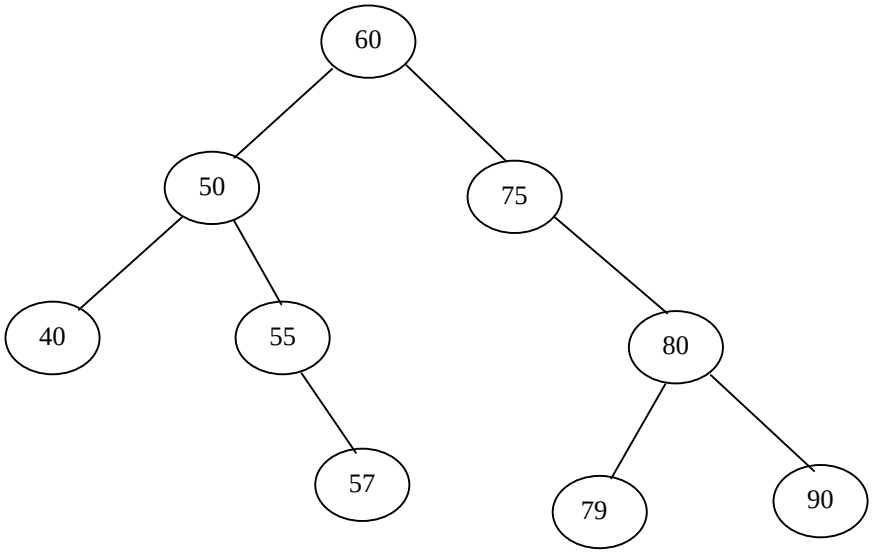
$$hash(n) = n \% 7$$

Does the following diagram illustrates the correct hash table resulting from performing the following operations?:

```
H.insert(40);
H.insert(11);
H.insert(7);
H.insert(3);
H.insert(9);
H.insert(1);
H.insert(16);
H.insert(0);
H.insert(48);
```



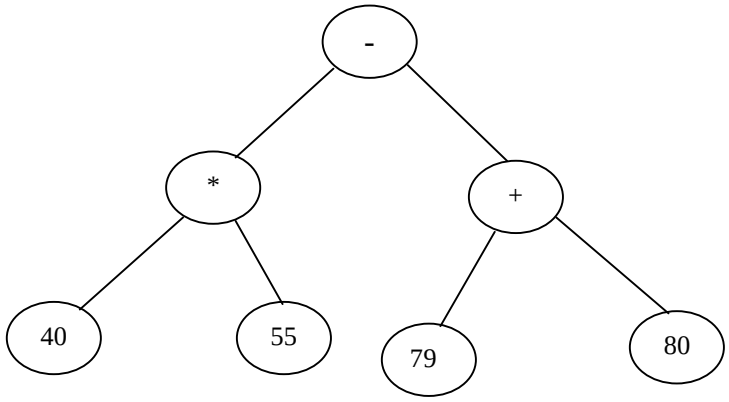
5. Consider the following Binary Search Tree.



- List the nodes in pre-order traversal.
- List the nodes in post-order traversal.
- List the nodes in in-order traversal.

5. An expression tree is a binary tree where:
   - Internal nodes represent a mathematical operator
   - Leaf nodes represent numbers

The following tree T, is an example of such tree.



Write a function, print_expression that prints the expression in the tree in a fully parenthesized form. For example, the result of this function on the above tree should be: (40 * 55) – (79 + 80). The prototype for function print_expression is a follows:

```
template <typename Object>
void ExpressionTree::print_expression(ostream& out);
```

6. Suppose that you have an implementation of a Binary Search Tree where the t_size data member indicating the size of the tree is not present, and the size of the tree must be computed "on-the-fly". Write a function size() that computes the size of the tree.(Hint: Use a recursive auxiliary function to traverse tree).

```
template <typename Object>
void BinarySearchTree::size() const{
```