Stephen Tambussi
6/4/2020
COEN 79

1)

```
//Q1
template <class Item>
binary_tree_node<Item>* tree_copy(const binary_tree_node <Item>* root_ptr)
{
    binary_tree_node<Item> *l_ptr;
    binary_tree_node<Item> *r_ptr;
    if(root_ptr == NULL) return NULL;
    else
    {
        l_ptr = tree_copy(root_ptr->left());
        r_ptr = tree_copy(toor_ptr->right());
        return new binary_tree_node<Item>(root_ptr->data(), l_ptr, r_ptr);
    }
}

template <class Item>
void bag <Item>::operator = (const bag<Item>& source)
{
    if(this == &source) return;
    tree_clear(root_ptr);
    root_ptr = tree_copy(source.root_ptr);
}
```

2)

```
template <class Item>
void bag<Item>::insert(const Item& entry)
{
    binary_tree_node<Item> *cursor = root_ptr;
    bool done = false;
    if(root_ptr == NULL)
    {
        root_ptr = new binary_tree_node<Item>(entry);
        return;
```

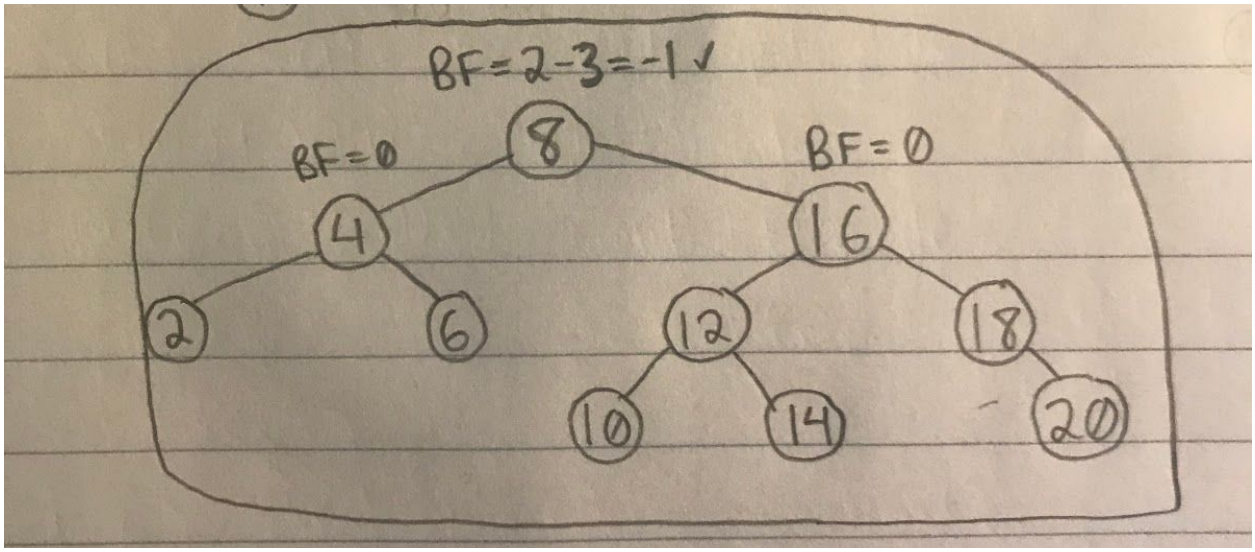
```
    }
    do
    {
        if(cursor->data() >= entry)
        {
            //left
            if(cursor->left() != NULL)
                cursor = cursor->left();
            else
            {
                cursor->left() = new binary_tree_node<Item>(entry);
                done = true;
            }

        }
        else
        {
            //right
            if(cursor->right() != NULL)
                cursor = cursor->right();
            else
            {
                cursor->right() = new binary_tree_node<Item>(entry);
                done = true;
            }

        }

    } while (!done);

}
```

3)

```
template <class Item>
binary_tree_node<Item>* left_right_rotation(binary_tree_node<Item>*&
parent)
{
    binary_tree_node<Item>* tmp1;
```

```
    tmp1 = parent->right();
    parent->set_right(tmp1->left());
    tmp1->set_left(parent);
    binary_tree_node<Item>* tmp2;
    tmp2 = tmp1->left();
    tmp1->set_left(tmp2->right());
    tmp2->set_right(tmp1);
    return tmp2;
}
```

4)



5)

```
template <class Item>
binary_tree_node<Item>* balance(binary_tree_node <Item>*& temp)
{
    if(diff(temp) >= 2)
    {
        if(diff(temp->left()) == -1)
            return balance(left_right_rotation(temp));
        else
            return balance(left_rotation(temp));
    }
    else if(diff(temp) <= -2)
    {
        if(diff(temp->right()) == 1)
```

```
            return balance(right_left_rotation(temp));
        else
            return balance(right_rotation(temp));
    }
}
```

6)

```
template <class Item>
void flip(binary_tree_node<Item>* root_ptr)
{
    if(root_ptr == NULL) return;
    else
    {
        binary_tree_node<Item>* temp;
        flip(root_ptr->left());
        flip(root_ptr->right());

        temp = root_ptr->left();
        root_ptr->left() = root_ptr->right();
        root_ptr->right() = temp;
    }

}
```