1. Part a)

Create a class called MyDegree. It should have two variables of type double. One called fahrenheit and the other celsius. It should have one constructor that takes in two variables, the first of type double the second of type String. Depending on the value of the String parameter (which should be either fahrenheit or celsius) set the corresponding variable to the value passed as a parameter. Compute the value for the other variable. (Hint: Fahrenheit to Celsius: f = (9/5)\*c +32, Celsius to Fahrenheit: c = (5/9)\*(f-32). Create 2 get methods, one for each variable. Create 2 set methods, one for each variable (don't forget to change the other variable from within each set method).

Part b)

Create a class called MyDate. It should have 3 variables of type int: named date, month, and year. It should have 2 variables of type MyDegree: named high and low. Create a constructor that takes in 3 parameters of type int. The constructor should call method isValid and pass it all three parameters. If the 3 parameter were all valid values, they should be set to the date month and year variables. If the isValid method returns false, some default date should be used instead (you choose what the default date value should be) and should also display the message to user saying that "The Entry is invalid. By default the date will be set to: *whatever you choose as default date*". Create a method isValid that will verify if the values passed are a valid date. Create set methods for the high and low variables. These 2 set methods should each take in 2 parameters, those 2 parameters should be passed to the constructor of class MyDegree and the resulting object created should be set to the variable (high or low). Create get methods for all 5 variables. The get methods for high and low should not return the MyDegree object, but instead should return the degree value of that object as an int. (Make sure to convert it to an int)

Part c)

Create a class called main which has a main method within it. The main class should ask the user for a date, month, and year. Create a MyDate object. Using the get methods of MyDate, verify with the user that the MyDate object has the correct date. Remember, you must get the values from the MyDate object and not use the variables that were passed to the MyDate constructor, since, if the date was not valid, a default date was used instead. If the user says "NO" when asked for confirmation, the user should be asked for another date. HINT: use a while loop.  If the user says "YES", the MyDate object should be stored in an array of type MyDate. HINT: create the array outside (ie. before) the while loop, otherwise it will become "out of scope" (ie. cease to exist) after the while loop. Create 4 more MyDate objects for the next 4 days in the calendar. HINT: use the isValid method from class MyDate to determine if these next dates are valid (for example, if the 1st date is June 30th, the 2nd date should be July 1st, not June 31st). Store these other MyDate objects in the array. Then, from within this main method, use the set methods of class MyDate to set the high and low variables. Make up values. Do not try to look up the high and low temperatures of those days. Lastly, print out to the console the high and low temperatures for all 5 days in Fahrenheit and then print out the high and low temperatures in Celsius.

An example output should look like this:

Fahrenheit:

2/2/2016 high 90 low 80

2/3/2016 high 88 low 78

2/4/2016 high 95 low 90

2/5/2016 high 85 low 80

2/6/2016 high 100 low 90

Celsius:

2/2/2016 high 32 low 26

2/3/2016 high 31 low 25

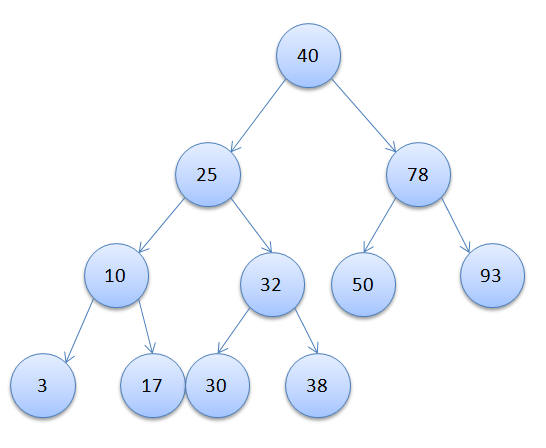
2/4/2016 high 35 low 32

2/5/2016 high 29 low 26

2/6/2016 high 37 low 32

## 2. What is a Binary Search Tree (BST)?

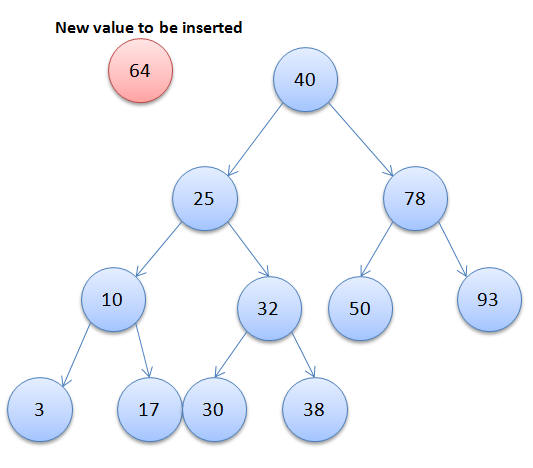
Binary Search Tree (BST) is a binary tree data structure with a special feature where in the value store at each node is greater than or equal to the value stored at its left sub child and lesser than the value stored at its right sub child. Let’s look at an example of a BST:

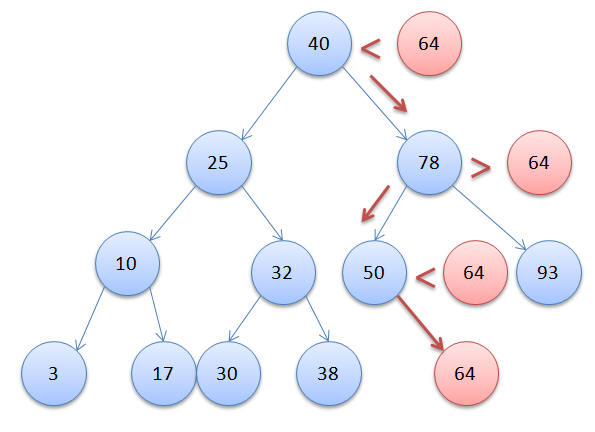


In the above example you can see that at each node the value in the left child is lesser than or equal to the value in the node and the value in the right child is greater than the value in the node.

## Building a Binary Search Tree (BST)

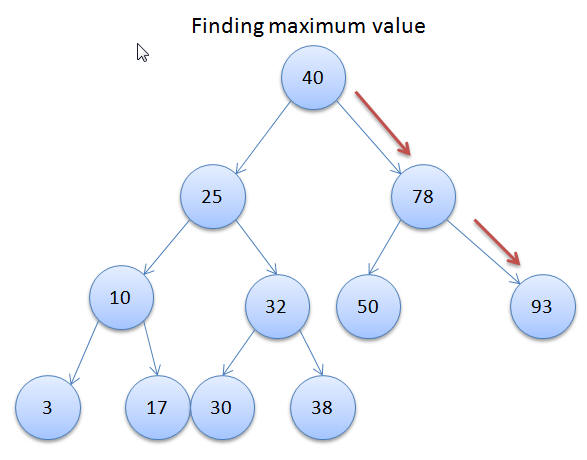
Now that we have seen how a BST looks, let me show you how one can build a BST and insert nodes into the tree by implementing the algorithm in Java. The basic idea is that at each node we compare with the value being inserted. If the value is lesser then we traverse through the left sub tree and if the value is greater we traverse through the right subtree. Suppose we have to insert the value 64 in the above BST, let’s look at the nodes traversed before its inserted at the right place:

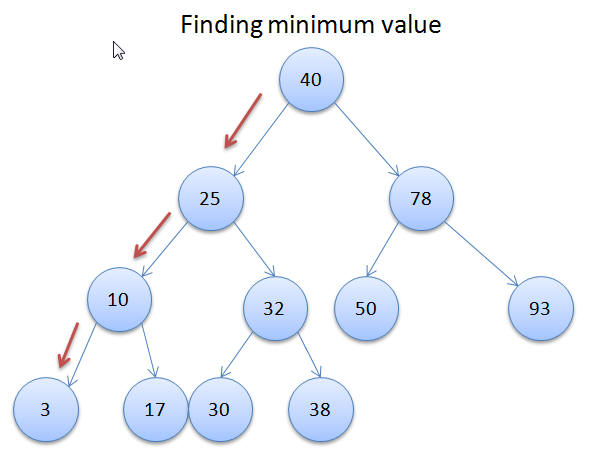




## Finding Maximum and Minimum Value in BST

If you have noticed in the above example that the leftmost node has the lowest value and the rightmost node has the highest value. This is due to the sorted nature of the tree.





Also, perform following operations:

* Searching a node
* Counting the total number of nodes

For inserting the nodes, following this order:

40, 25, 78, 10, 3, 17, 32, 30, 38, 78, 50, 93