SML Homework

Jen McCall, John Mulcahy

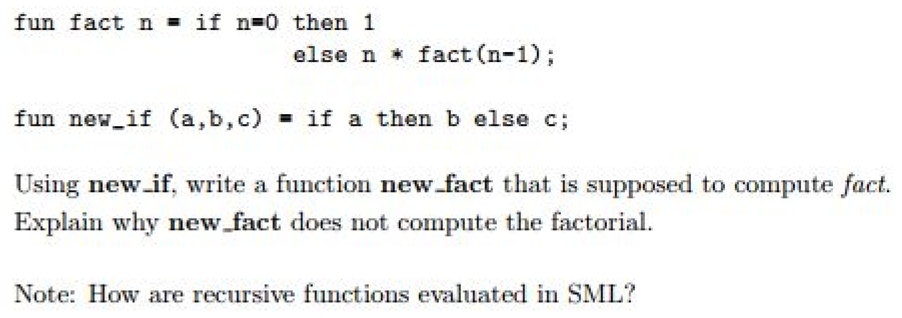
CS 361

1. What are the types of the following expressions?
   - [(1,5), (2,3), (5,6)];
     - `(int * int) list`
   - fun f(x:real) = true;
     - `real -> bool`
   - map f;
     - `‘a list -> ‘b list`
2. Provide expressions of the following types:
   - int * bool
     - `(4, true)`
   - int list * bool
     - `([5,8], true)`
   - int * real -> bool list
     - `(1,3.4)-> [true, true]`

3. Write the following SML functions:

Write a recursive function that computes 2^n for $n \geq 0$.

```
f(n) = 2^n
int -> int
fun exp(n) = if n=0 then
     1
 else exp(n-1) * 2;
```



```
fun fact n = if n=0 then 1
                 else n * fact(n-1);

fun new_if (a,b,c) = if a then b else c;
```

Using **new_if**, write a function **new_fact** that is supposed to compute *fact*.
Explain why **new_fact** does not compute the factorial.

Note: How are recursive functions evaluated in SML?

```
fun new_fact(a,b,c) = if a then b else c * fact(a, b,
c-1);

new_fact(1,1,5)
```

This new_fact doesn’t run because our base case for factorial requires us to check if the number is equal to zero. In our ideal function, we are checking the value of a and then doing b, but this doesn’t work if a itself relies on the value of c. We can’t pass/check the value of c through a, so our base case in new_fact would never properly work. Furthermore, this would also rely on b always being equal to 1. If that's the case, we should consider getting rid of the variable b and just requiring 1, it makes no sense to have the base case in the function parameters like that.

Define a function circumference that computes the circumference of a circle with respect to its radius. Use pi from the Math library.

```
fun circumference(r:real) =
    2.0*Math.pi*r;

circumference(14.0)
```

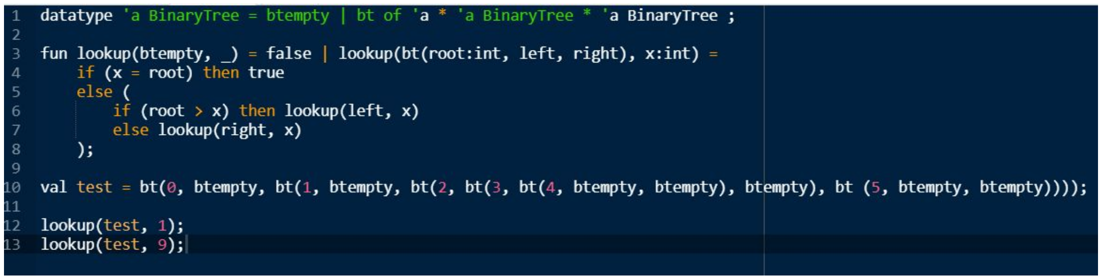
How to use map to add 3 to each elements of a list

```
val L = [1, 1, 2, 3];
fun addThree(x) = x + 3;
map addThree L;
```

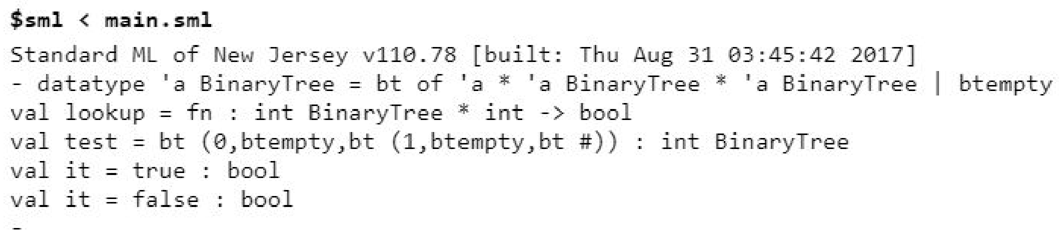
Write a function *move* that transforms a list $[a_1, ..., a_n]$ into a list $[a_2, ..., a_n, a_1]$.

```
fun addOne(L) =
    if L = [] then []
    else
        tl(L) @ [hd(L)];
```

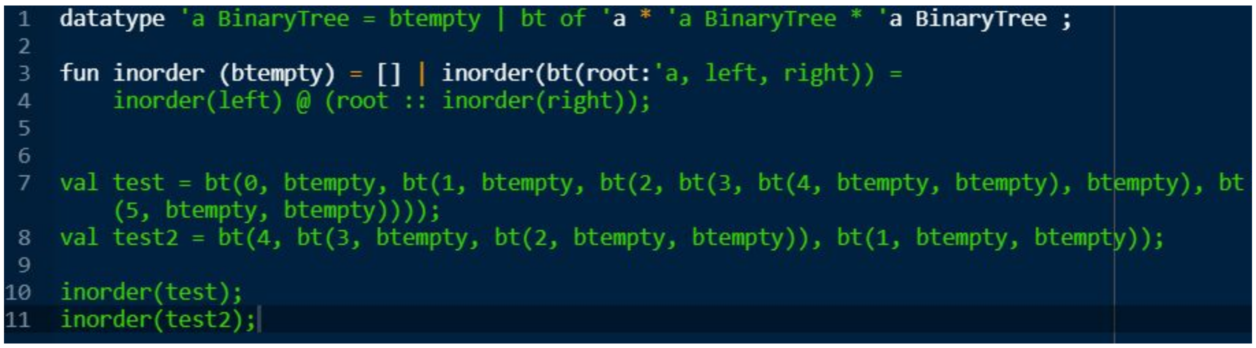
4. Implement the datatype BinaryTree and all the functions that are provided in the lecture notes: lookup, inorder, preorder, postorde, left_subtree, right_subtree and label. Provide screenshots to show that your code is correct. Provide 2 tests for each function.



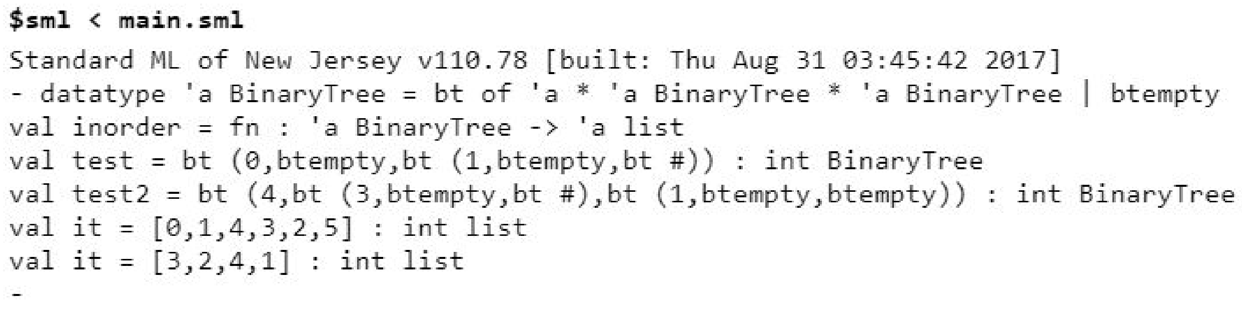
```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

fun lookup(btempty, _) = false | lookup(bt(root:int, left, right), x:int) =
    if (x = root) then true
    else (
        if (root > x) then lookup(left, x)
        else lookup(right, x)
    );

val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt (5, btempty, btempty))));

lookup(test, 1);
lookup(test, 9);
```



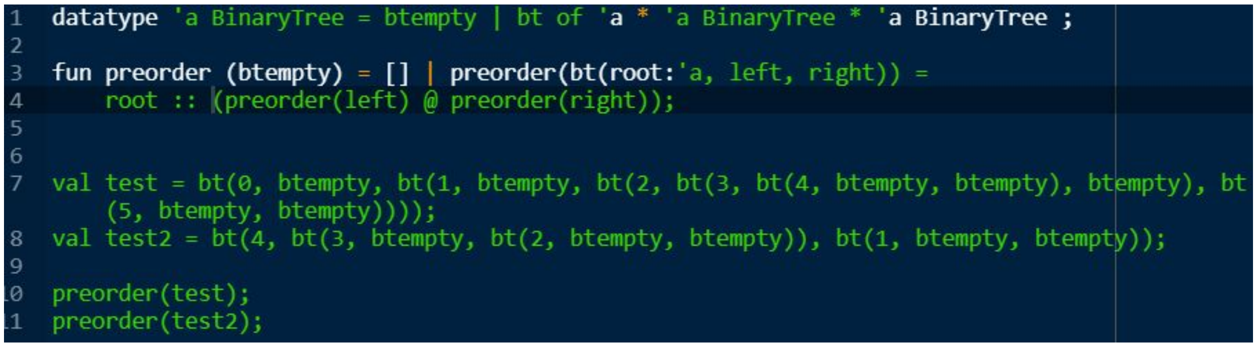
```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
val lookup = fn : int BinaryTree * int -> bool
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val it = true : bool
val it = false : bool
-
```



```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

fun inorder (btempty) = [] | inorder(bt(root:'a, left, right)) =
    inorder(left) @ (root :: inorder(right));


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

inorder(test);
inorder(test2);
```

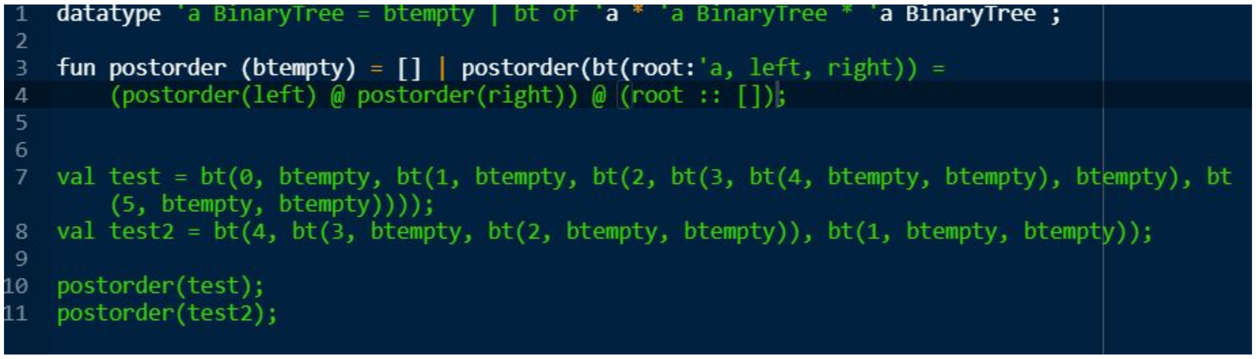


```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
val inorder = fn : 'a BinaryTree -> 'a list
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = [0,1,4,3,2,5] : int list
val it = [3,2,4,1] : int list
-
```



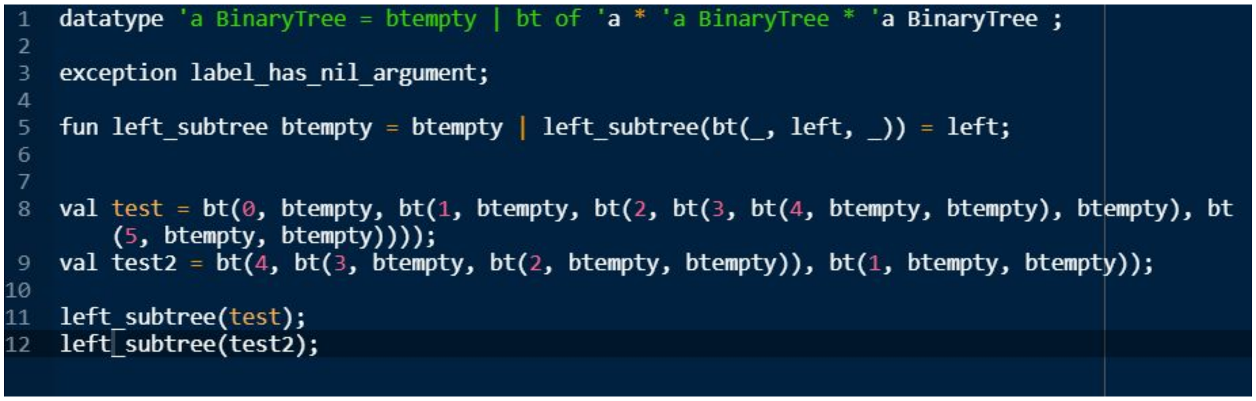
```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

fun preorder (btempty) = [] | preorder(bt(root:'a, left, right)) =
    root :: (preorder(left) @ preorder(right));


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt
    (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

preorder(test);
preorder(test2);
```

```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
val preorder = fn : 'a BinaryTree -> 'a list
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = [0,1,2,3,4,5] : int list
val it = [4,3,2,1] : int list
-
```



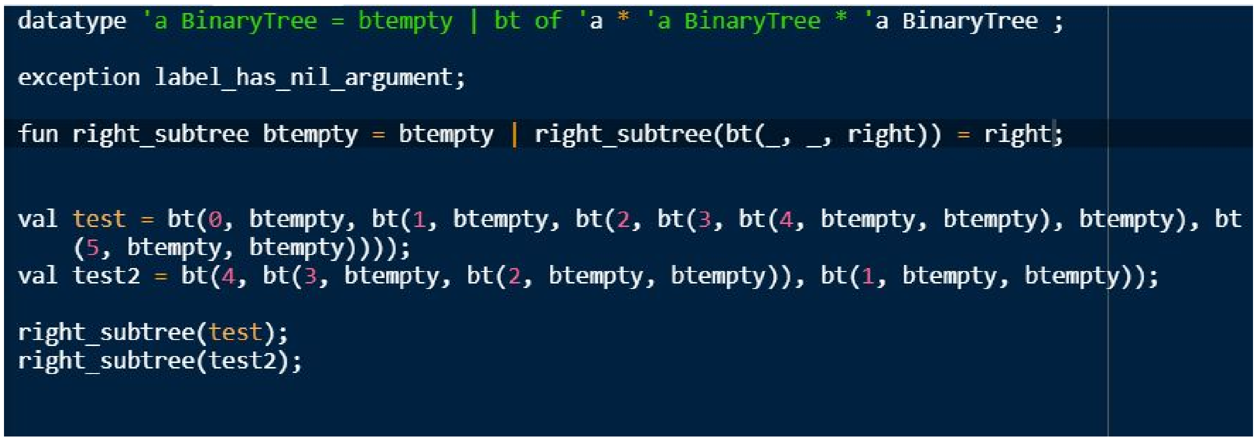
```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

fun postorder (btempty) = [] | postorder(bt(root:'a, left, right)) =
    (postorder(left) @ postorder(right)) @ (root :: []);


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt
    (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

postorder(test);
postorder(test2);
```

```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
val postorder = fn : 'a BinaryTree -> 'a list
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = [4,3,5,2,1,0] : int list
val it = [2,3,1,4] : int list
-
```



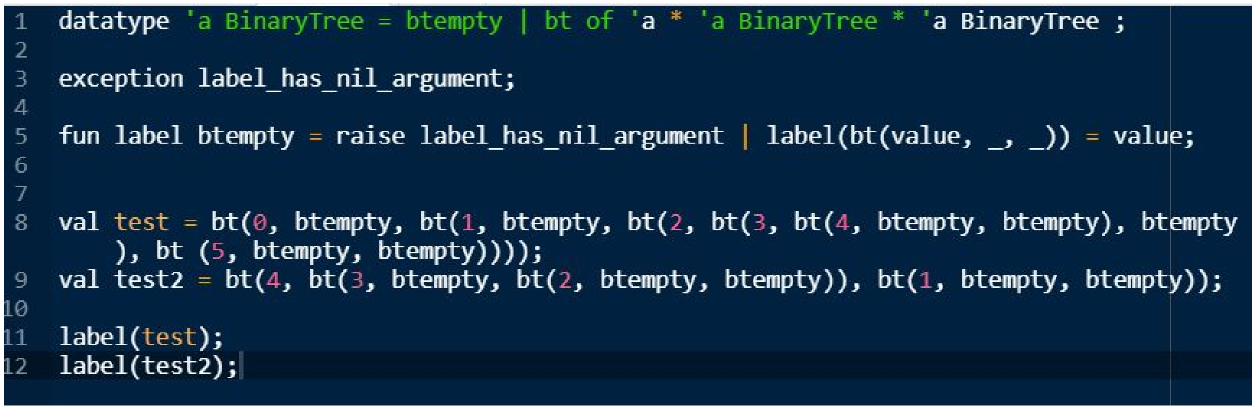
```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

exception label_has_nil_argument;

fun left_subtree btempty = btempty | left_subtree(bt(_, left, _)) = left;


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt
    (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

left_subtree(test);
left_subtree(test2);
```

```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
exception label_has_nil_argument
val left_subtree = fn : 'a BinaryTree -> 'a BinaryTree
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = btempty : int BinaryTree
val it = bt (3,btempty,bt (2,btempty,btempty)) : int BinaryTree
-
```

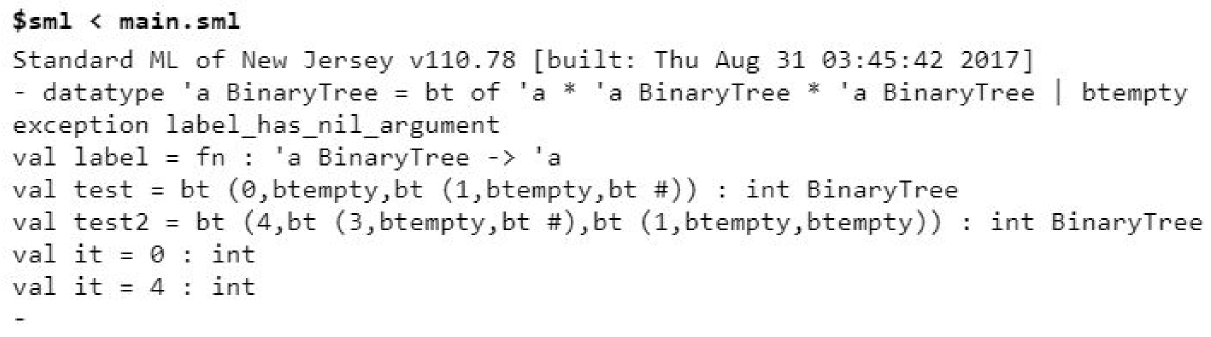


```
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

exception label_has_nil_argument;

fun right_subtree btempty = btempty | right_subtree(bt(_, _, right)) = right;


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty), bt
    (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

right_subtree(test);
right_subtree(test2);
```

```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
exception label_has_nil_argument
val right_subtree = fn : 'a BinaryTree -> 'a BinaryTree
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = bt (1,btempty,bt (2,bt #,bt #)) : int BinaryTree
val it = bt (1,btempty,btempty) : int BinaryTree
-
```



```sml
datatype 'a BinaryTree = btempty | bt of 'a * 'a BinaryTree * 'a BinaryTree ;

exception label_has_nil_argument;

fun label btempty = raise label_has_nil_argument | label(bt(value, _, _)) = value;


val test = bt(0, btempty, bt(1, btempty, bt(2, bt(3, bt(4, btempty, btempty), btempty
    ), bt (5, btempty, btempty))));
val test2 = bt(4, bt(3, btempty, bt(2, btempty, btempty)), bt(1, btempty, btempty));

label(test);
label(test2);
```



```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- datatype 'a BinaryTree = bt of 'a * 'a BinaryTree * 'a BinaryTree | btempty
exception label_has_nil_argument
val label = fn : 'a BinaryTree -> 'a
val test = bt (0,btempty,bt (1,btempty,bt #)) : int BinaryTree
val test2 = bt (4,bt (3,btempty,bt #),bt (1,btempty,btempty)) : int BinaryTree
val it = 0 : int
val it = 4 : int
-
```