1 Bit Warm up

1.1 Why 2's complement?

The mantra of 2's complement is "flip and add one." However, to understand exactly why we choose this method to represent negative numbers it helps to understand another approach called 1's complement.

In 1's complement we flip all the bits to get the inverse sign of a number (no addition needed!) For example, using 8 bit numbers $5 = 0b00000101$, so $-5 = 0b11111010$ and $43 = 0b00010111$, so $-43 = 0b11101110$. (Note: The prefix $0b$ indicated the number should be interpreted in binary)

(a) Translate the following numbers in both 1 and 2's complement.
You should assume all numbers are 8 bits long.

Number	1′s complement	2′s complement
-26		
-2		
-1		
-0		
0		
1		

(b) So far both of these seem like valid representations of negative numbers, however let's see what happens when we do arithmetic operations with them. Calculate the following values using 1 and 2's complement numbers. (Note: In 1's complement, if the addition of the most significant requires a carry, you must "wrap around" by adding 1 to the least significant bit)

Number	1′s complement	2′s complement
$3+5$	$0b0000011$ $+0000101$ $= 0001000$	$0b0000011$ $+0000101$ $= 0001000$
$-1+-6$	$+$ _____ $=$	$+$ _____ $=$
$-2+5$	$+$ _____ $=$	$+$ _____ $=$

(c) 1's complement produces the mathematically incorrect value for one of these expressions. Why? Why does this error not occur in 2's complement?

2 ASCII

ASCII is a method of representing text characters in binary using 8 bits per character. Below is a representation of all ASCII values and how they translate to binary.

USASCII code chart

Bits $b_7 b_6 b_5$ →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

To read this chart we append the value in the column of each character to the value of its row. For example, "A"'s column is 100, and it's row is 0001 so $A = 01000001 = 65$.

(a) What is the relationship between capital and lowercase letters in ASCII? What is the relationship in terms of binary representation?

(b) Using the information from this chart and bit operation(s), implement toUpperCase by filling in the blank, which will print the passed in String *s* with all it's characters in upper case. For example:

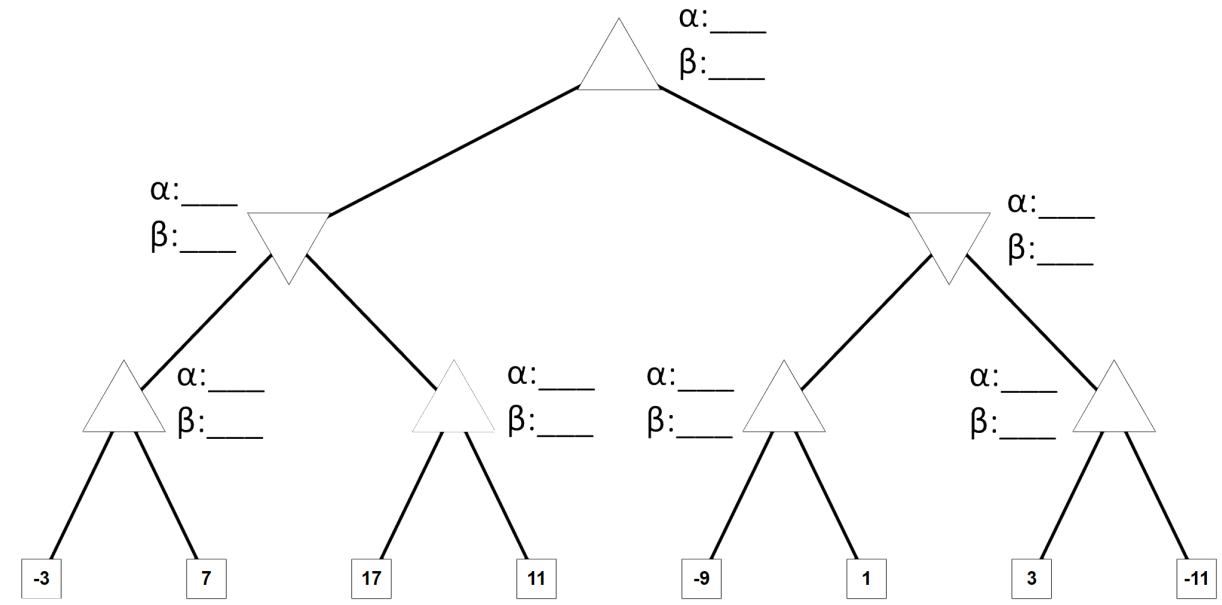
```
String s = "gobears"
>>>toUpperCase(s)
"GOBEARS"
s = "GoBeArS"
>>>toUpperCase(s)
"GOBEARS"

public String toUpperCase(String s) {
        for (int i = 0; i < s.length; i++) {

                System.out.print(_____________________________________);
        }
}
```

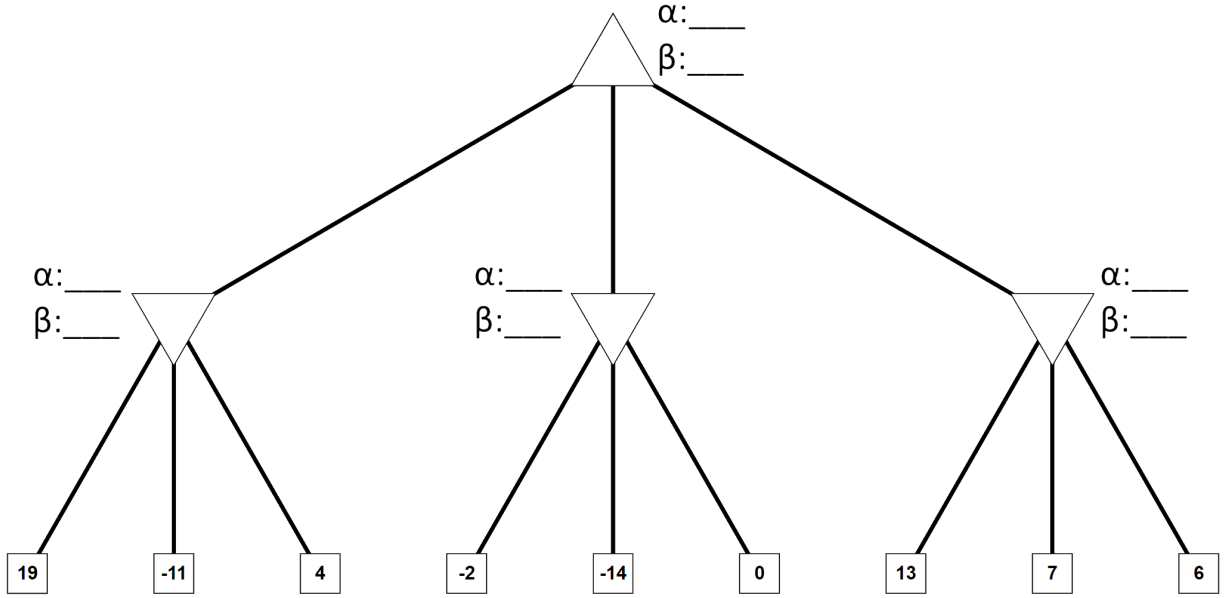
(c) Now fill in the blank to implement toLowerCase using bit operation(s), which prints a string converted to lowercase. For example:

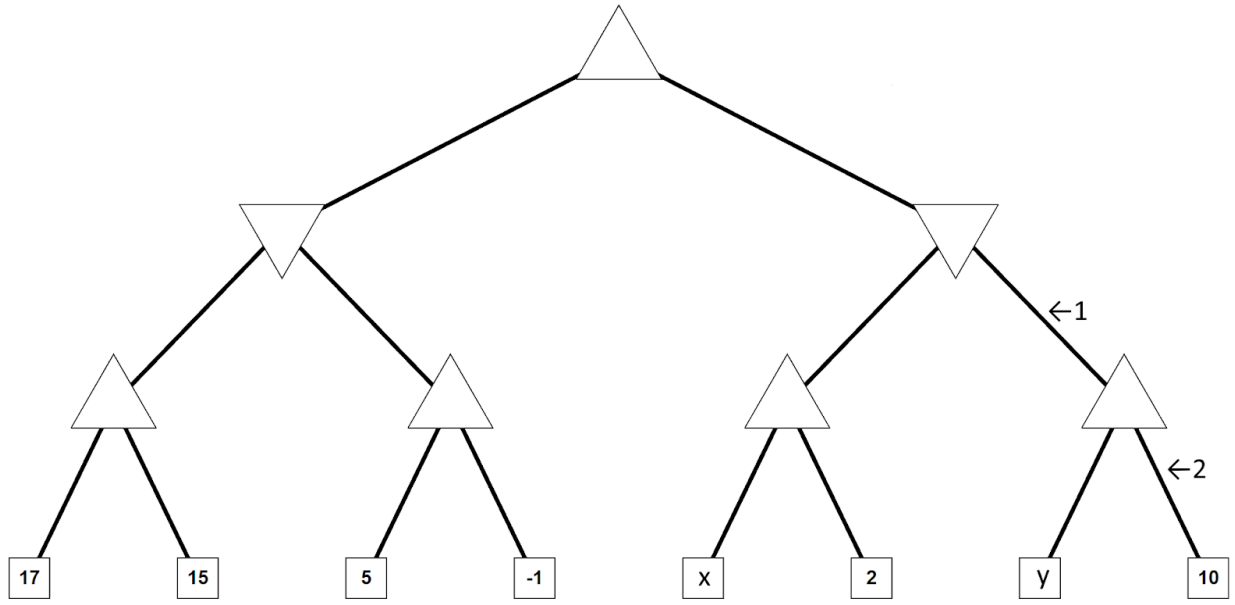
```
String s = "GOBEARS"
>>>toLowerCase(s)
"gobears"
s = "GoBeArS"
>>>toLowerCase(s)
"gobears"

public String toLowerCase(String s) {
        for (int i = 0; i < s.length; i++) {

                System.out.print(_____________________________________);
        }
}
```

3 Game Trees

Evaluate the following game trees using alpha beta pruning. Cross out any branches that would be pruned. Assume pruning occurs when $\beta \leq \alpha$.
For all parts of this question, use ∞ or $-\infty$ to indicate positive or negative infinity if necessary.







(a) Given the game tree above, define bounds on x such that the branch labeled 1 will be pruned.

$$\_\_\_\_\_ < x \leq \_\_\_\_\_$$

(b) Assuming the condition you specified in part *a* is not met, Determine a relationship between x and y that will result in the branch labeled 2 to be pruned.

(c) Define ranges for x and y such that the value 10 will propagate up to the top of the game tree. Assume $y \neq 10$ and $x \neq 10$.

$$\_\_\_\_\_ < x < \_\_\_\_\_ \qquad \_\_\_\_\_ < y < \_\_\_\_\_$$

4 Asymptotic Approach

Give a tight asymptotic runtime bound on `foo(n)`.

```
1  int foo(int n) {
2      if (n == 0) {
3          return 0;
4      }
5      baz(n);
6      return foo(n/3) + foo(n/3) + foo(n/3);
7  }
8
9  int baz(int n) {
10     for (int i = 0; i < n; i += 1) {
11         System.out.println("Help me! I'm trapped in a loop");
12     }
13     return n;
14 }
```

5 Asymptotic Potpourri

For each of the following code snippets, give a tight asymptotic runtime bound with respect to n.

(a)
```
public void mystery1(int n) {
    for (int i = 0; i < n; i += 1) {
        for (int j = 0; j < n; j += 1) {
            i = i * 2;
            j = j * 2;
        }
    }
}
```

(b)
```
public void mystery2(int n) {
    for (int i = n; i > 0; i = i / 2) {
        for (int j = 0; j < i * 2; j += 1) {
            System.out.println("Hello World");
        }
    }
}
```

(c)
```
public void mystery3(int n) {
    for (int i = n; i > 0; i = i / 2) {
        for (int j = 0; j < i * i; j += 1) {
            System.out.println("Hello World");
        }
    }
}
```

(d)
```
public void mystery4(int n) {
   int i = 1, s = 1;
   while (s <= n) {
        i += 1
        s = s + i;
        System.out.println(s);
    }
}
```

6 Joe Loves This Question

Determine which data structure(s) are best suited for the scenarios below in terms of performance, taking into account the specific types of inputs listed in each problem. Your descriptions of the data structure(s) chosen should be **brief** but sufficiently detailed so that the runtime is unambiguous Give the worst-case runtime bound of your solution in Big-Theta notation. (Modified from Summer 2016)

a. Given a large text of N words, find the number of occurrences of each unique word.

Data Structures:

Usage:

Runtime:

b. You are given a collection of N reviews. Each review has a date, author name, number of stars, and the text contents of the review. We would like to determine the number of reviews within a certain date-time range. Optimize for both query and construction time.

Data Structures:

Usage:

Runtime (construction): Runtime (query):

c. You are given N images of size 256×256, each represented as a 2-D Array (i.e. `int[][]`). Each image has associated with it a saturation value, which can be calculated from the pixels. Support the following operations: `add(int[][] img)`, `getAllImgWithSaturation(int saturation)`, and `remove(int[][] img)` Data Structures:

Usage:

Runtime (construction):

7 Hashing Mechanics

Suppose we insert the following words into an initially empty hash table, in this order: **galumphing, frumious, slithy, borogroves, mome, bandersnatch**. Assume that the hash code of a `String` is just its length (note that this is not actually the hash code for `String`s in Java). Use separate chaining to resolve collisions. Assume 4 is the initial size of the hash table's internal array, and double this array's size when the load factor is equal to 1. Illustrate this hash table below with a box-and-pointer diagram.

8 Hash Code Design

Describe a potential problem with each of the following:

1. An implementation of the hashCode method of the `String` class that simply returns the length of the string (i.e. the hash code used in the previous problem).

2. An implementation of the hashCode method of the String class that simply returns a random number each time.

3. Overriding the equals method of a class without overriding the hashCode method.

4. Overriding the hashCode method of a class without overriding the equals method.

5. Modifying an object after inserting it into a HashSet.

9 Assorted Heap Questions

(a) Describe a way to modify the usual min heap implementation so that finding the maximum element takes constant time without incurring more than a constant amount of additional time and space for the other operations.

(b) In class, we looked at one way of implementing a priority queue: the binary min heap. Recall that a binary min heap is a nearly complete binary tree such that any node is smaller than both of its children. There is a natural generalization of this idea called a d-ary min heap. This is also a nearly complete tree where every node is smaller than all of its children. But instead of every node having two children, every node has d children for some fixed constant d.

i. Describe how to insert a new element into a d-ary heap (this should be very similar to the binary heap case). What is the running time in terms of d and n (the number of elements)?

ii. What is the running time of finding the minimum element in a d-ary heap with n nodes in terms of d and n?

iii. Describe how to remove the minimum element from a d-ary heap (this should be very similar to the binary heap case). What is the running time in terms of d and n?

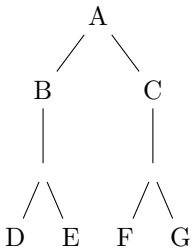
(c) Suppose a value in a min heap were changed. To recreate the heap to reflect the modified value, would you bubble the value up, bubble it down, both, or neither? Briefly defend your answer.

10 Hidden Message

(a) Given the post-order sequence (G A U F H) and in-order sequence (G U A H F), construct the tree and find the pre-order sequence.

(b) Given the post-order sequence (T R D I E) and pre-order sequence (E D T R I), construct the tree and find the in-order sequence. For this question you can assume the tree is full (a full tree means that each node will always have two children).

(c) Suppose we are searching for nodes in the following tree:



To search this tree, we may use either of the following two traversal methods: preorder depth-first and level-order. Assume that both traversals will break ties by going left. Which method will require us to look at the fewest nodes, assuming we're looking for node D? What about for nodes C and G? Keep in mind the answer may be a tie.

11 Your Node is my Node

We've seen how a simple, encapsulated `Node` class can be used to power very versatile data structures like `LinkedLists` and `BinarySearchTrees`. Fill in the method, `treeToList`, below such that it destructively mutates the nodes of the tree into a linked list structure while still maintaining its relative ordering.

```
public class BinarySearchTree<T> {

    private Node root;

    private class Node {
        public T value;
        public Node left;
        public Node right;
    }

    public Node treeToList(Node root) {
        if (root == null) {
            return null;
        }
        Node leftList = _________________;
        Node rightList =_________________;
        link(_______________, _______________);
        leftList = extend(_____________,_____________);
        leftList = extend(_____________,_____________);
        return leftList;
    }

    /* Extends the leftList with the elements of the rightList.
       Assumes that the given Nodes represent circular DLLs */
    private Node extend(Node leftList, Node rightList) {
        if (leftList == null) {
            return _____________;
        }
        if (rightList == null) {
            return _____________;
        }
        Node endOfLeft = leftList.left;
        Node endOfRight = rightList.left;
        _______________________________;
        _______________________________;
        return leftList;
    }

    /* Makes b the next node of a, and a the previous node of b */
    private void link(Node a, Node b) {
        a.right = b;
        b.left = a;
    }
}
```