King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering
Information and Computer Science Department
First Semester 181 (2018/2019)

ICS 202 – Data Structures
Major Exam 2
Sunday, November 4th, 2018
Time: 90 minutes

Name: _____KEY_____

ID# | | | | | | | | | |

	Question #	Max Marks	Marks Obtained
Section 01 Dr. Sami	1	20	
	2	20	
Section 02 Dr. Irfan	3	30	
	4	30	
	Total	100	

Instructions

1. Write your name and ID in the respective boxes above and circle your section.
2. This exam consists of 8 pages, including this page and a "Quick Reference Sheet" at the end.
3. This exam consists of 4 questions. You have to answer all the 4 questions.
4. The exam is closed book and closed notes. No calculators or any helping aids are allowed.
5. Make sure you turn off your mobile phone and keep it in your pocket if you have one.
6. The questions are not equally weighed.
7. The maximum number of points for this exam is 100.
8. You have exactly 90 minutes to finish the exam.
9. Make sure your answers are readable.
10. If there is no space on the front of the page, feel free to use the back of the page. Make sure you indicate this in order not to miss grading it.

Q.1 (20 points) [Stacks and Queues]:

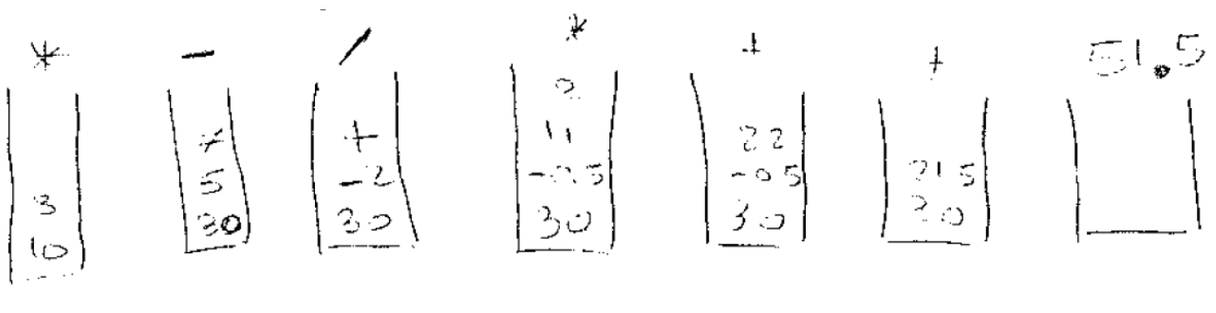
(a) [10 points] Consider the Java predefined class ***Stack*** (*java.util.Stack*). Write the code of a new method in that class called ***reverse*** which reverses the content of the current stack (*this*). That is, after the call to reverse the elements at the top of the stack will be on the bottom, etc. You can use any other method in the ***Stack*** class (see reminder sheet at the end of the exam).

```
public void reverse()
     {
     Stack<T> stack1 = new Stack<T>();
     while (!this.isEmpty())
          stack1.push(this.pop());

     Stack<T> stack2 = new Stack<T>();
     while (!stack1.isEmpty())
          stack2.push(stack1.pop());

     while(!stack2.isEmpty())
          this.push(stack2.pop());
     }
```

(b) (10 points) Consider the following postfix expression:

10 3 * 5 7 – 4 / 11 2 * + +

(i) [7 points] Evaluate the postfix expression using a stack. Show the contents of the stack after each operation.



(ii) [3 points] Show the unambiguous infix equivalent of the above expression (you can use parentheses).

(10*3) + ((5-7)/4) + (11*2)

Q2. (20 points) Recursion and Analysis of Recursion

(a) [10 points] Write a recursive method to count the number of leaves in a binary search tree. The method should return the count as an integer.

```
public void countLeaves(BSTNode<T> n){
  if (n == null)
       return 0;
  if(n.left == null && n.right== null)
       return 1;
  return countLeaves(n.left) + countLeaves(n.right);
}
```

(b) [10 points] Write a recurrence relation for the following code fragments. Represent the time complexity T(n) of this method in terms of n. Solve the recurrence relation, what is the Big-Oh complexity of the method? You can assume n to a power of 2. Hint: Base the recurrence on counting the additions.

```
int mystery(int n) {
     if(n==1)
          return n;
     return mystery(n/2) + mystery(n/2);
}
```

$T(n) = 0$ for $n = 1$

$T(n) = 2T(n/2) + 1$ for $n > 1$

$= 4T(n/4) + 3$

$= 2^kT(n/2^k) + 2^k - 1$

$2^k = n;\ k = \log_2 n$

$2^kT(n/2^k) + 2^k - 1 = 0 + 2^{\log_2 n} - 1 =$ **n-1**

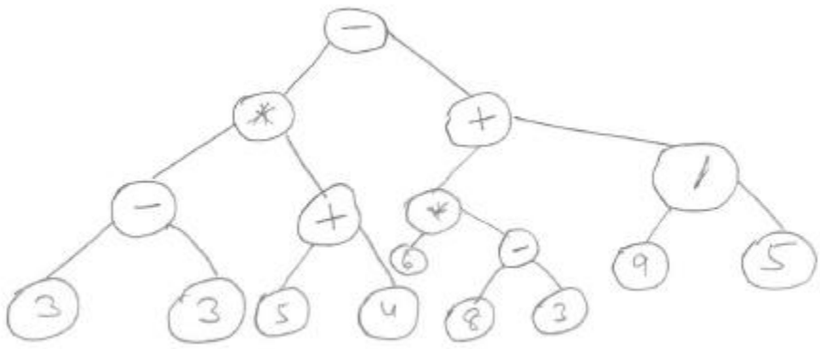
O(n)

Q.3 (30 points) Expression Trees and Heaps

(a) [12 points] Represent the following infix expression as an expression tree. Convert the expression into prefix and postfix form using the traversals on the expression tree.

```
((3-3)*(5+4)-(6*(8-3)+(9/5)))
```

Expression tree:



Prefix expression:

```
- * - 3 3 + 5 4 + * 6 - 8 3 / 9 5
```

Postfix expression:

```
3 3 - 5 4 + * 6 8 3 - * 9 5 / + -
```

(b) [6 points] For the following min-heap, show the resultant min-heap after deleting element 21.

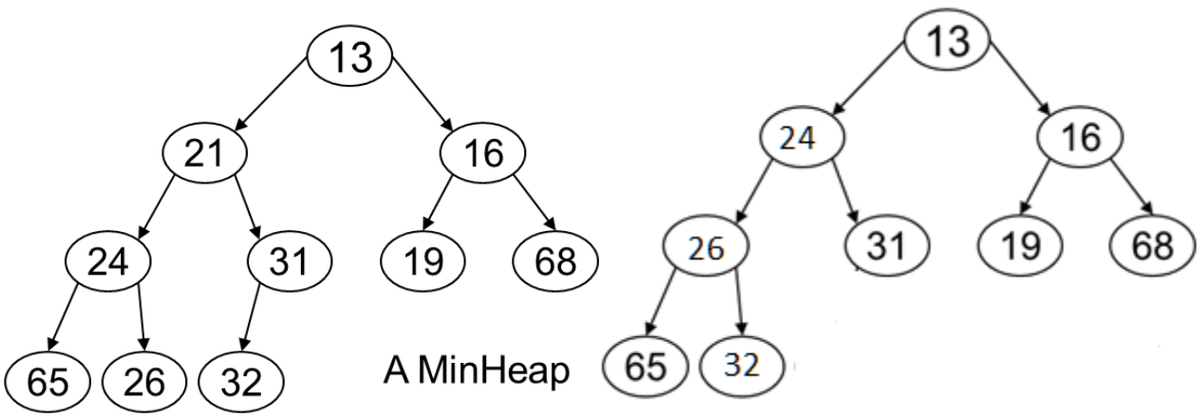
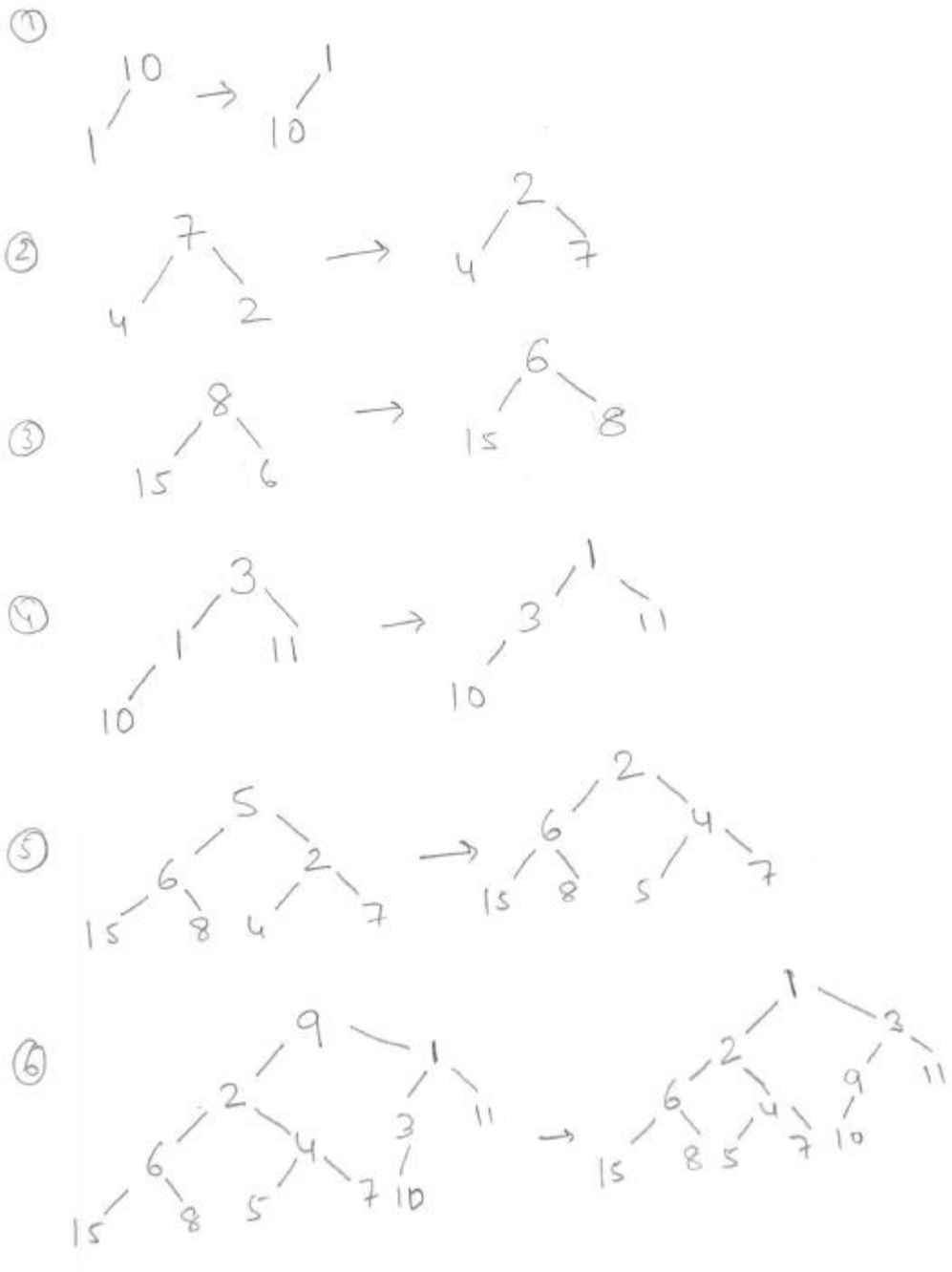


Figure 1: Solution

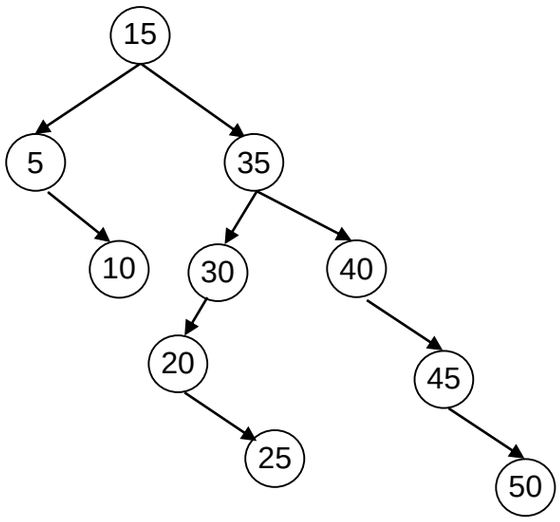
(c) [12 points] Create an min-heap for the following array elements using the bottom-up approach (Floyd's algorithm). Show the steps by drawing the heap at every step. Note: The heap at every step will not necessarily be a valid min-heap but it should finally become a valid min-heap.

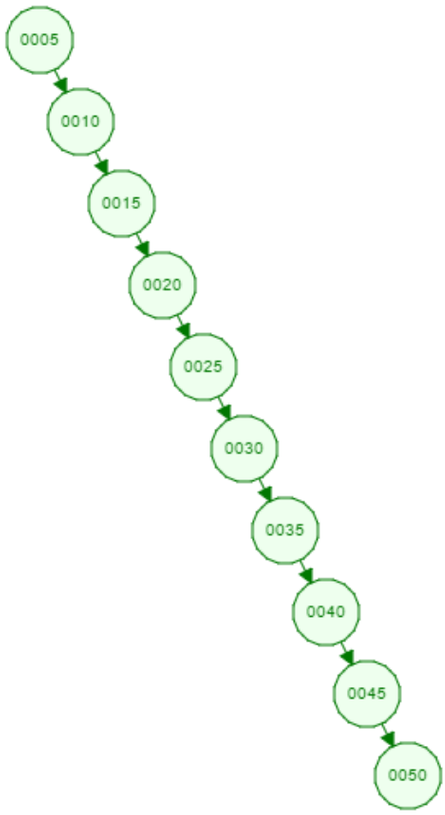
Input Array: `9 5 3 8 7 10 11 15 6 4 2 1`

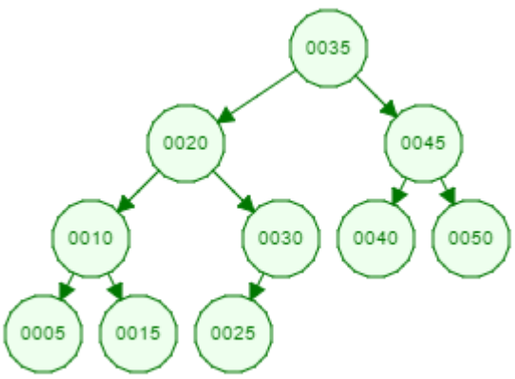


Q.4 (30 points) [Balanced and AVL Trees]

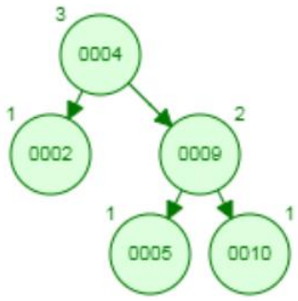
(a) [15 points] Apply the DSW Algorithm on the following binary search tree. Show the tree after the first step (CreateBackbone) and then in its final state (after CreatePerfectTree)

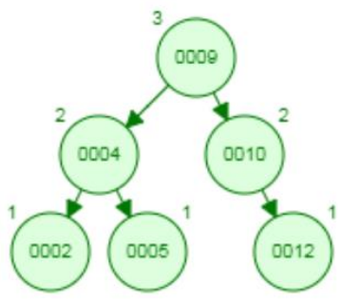


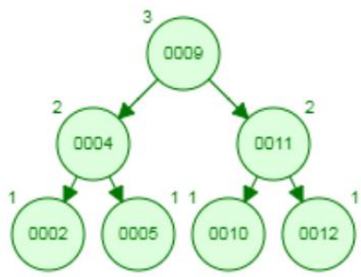

0005
0010
0015
0020
0025
0030
0035
0040
0045
0050


0035
0020
0045
0010
0030
0040
0050
0005
0015
0025

(b) (15 points) Insert these keys into an empty AVL-tree: 2, 5, 4, 9, 10, 12, 11







Quick Reference Sheet

```
public class Stack<T> {
     private …; // array or linked list
   public Stack();
   public Stack(int n);
   public void clear();
   public boolean isEmpty();
   public T topEl();
   public T pop();
   public void push(T el);
   public String toString();
}
```

```
public class Queue<T> {
     private …; // array or linked list
   public Queue();
   public void clear();
   public boolean isEmpty();
   public T firstEl();
   public T dequeue();
   public void enqueue(T el);
   public String toString();
}
```

$$\sum_{k=1}^{n}\frac{1}{k}=1+\frac{1}{2}+\ldots+\frac{1}{n}\approx\ln n$$

$$\sum_{k=1}^{n}\lg k\approx n\lg n$$

$$\sum_{k=0}^{n-1}c=cn.$$

$$\sum_{k=0}^{n-1}\frac{1}{2^k}=2-\frac{1}{2^{n-1}}$$

$$\sum_{k=1}^{n}k^2=\frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=0}^{n}k(k+1)=\frac{n(n+1)(n+2)}{3}$$

$$\sum_{i=m}^{n}c=c\left(\sum_{i=m}^{n}1\right)=c.(n-m+1)$$

$$\sum_{i=1}^{n}i=\frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n}i^2=\frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n}i^3=\left(\frac{n(n+1)}{2}\right)^2$$

$$\sum_{i=0}^{n}a^i=\frac{a^{n+1}-1}{a-1},a\neq 1$$

$$\sum_{k=0}^{n-1}2^k=2^n-1$$

$$\sum_{k=0}^{\infty}x^k=\frac{1}{1-x}\qquad \text{if } x<1$$

$$\log_b a=\frac{\ln a}{\ln b}\ ,\ \log ab=\log a+\log b$$

$$\log\frac{a}{b}=\log a-\log b\ ,\ a^{\log_a b}=b$$

$$\left(a^b\right)^c=(a^c)^b=a^{bc}$$