King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering
Information and Computer Science Department
First Semester 121 (2012/2013)

ICS 202 – Data Structures
Final Exam
Wednesday, January 2nd, 2013
Time: 120 minutes

Name: ____

ID#									

Section 01 Dr. Wasfi 10-10:50am
Section 02 Dr. Sami 9-9:50am

Question #	Max Marks	Marks Obtained
1	30	
2	20	
3	15	
4	15	
5	20	
Total	100	

Instructions

1. Write your name and ID in the respective boxes above and circle your section.
2. This exam consists of 10 pages, including this page, plus one additional reference sheet, containing 5 questions.
3. You have to answer all 5 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 120 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 [30 points] Multiple Choice Questions: Mark the best answer for each question below. Note: only one choice should be chosen.

1. Consider the following code segment

   ```
   sum = 0;
   for (i=1; i<n; i*=2)
     for (j=1; j<=i; j++)
       for (k=1; k<=j; k++)
          sum++; // Statement 1
   ```

   The number of times Statement 1 is executed, assuming $n > 1$ is a power of 2, is equal to

   a. n^2
   [✓ b. $\frac{n^2}{6}+\frac{n}{2}-\frac{2}{3}$]
   c. $\frac{1}{2} n \log n$
   d. $10n - 19$
   e. none of the above.

2. Consider the following method

   ```
   1) public void final(int [] A, int n) {
   2)   if (n == 1) {
   3)     System.out.print(A[n-1]+",");
   4)     return;
   5)   }
   6)   final(A, n/2);
   7)   System.out.print(A[n-1]+",");
   8)   final(A, n/2);
   9)   System.out.print(A[n-2]+",");
   10)  }
   ```

   The number of times the print statements in lines 3 and 7 are executed, assuming n is a power of 2, is equal to

   [✓ a. $3n - 2$]
   b. $2^{n+1} - 1$
   c. $2n - 1$
   d. $2^n - 1$
   e. none of the above.

3. The method `final` in Question 2 when called on A=[5,4,3,2,1] and n = 4 outputs

   a. an exception for an illegal index value for array A.
   b. 1,2,1,1,4,1,2,1,1,3,
   c. 5,3,5,4,1,5,3,5,4,2,
   [✓ d. 5,4,5,5,2,5,4,5,5,3,]
   e. none of the above.

4. The most efficient data structure in the worst case, among the following, in the cost of searching is

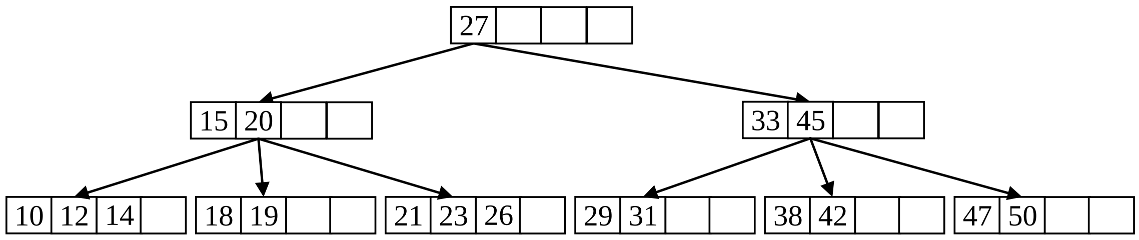
   a. an open-addressing hash table.
   b. a binary search tree.
   [✓ c. a sorted array.]
   d. a queue.
   e. both answers b and c as they are asymptotically equal.

5. The following statement is not true regarding arrays vs. singly linked lists of size $n > 1$.
    - a. Removing the last element in an array is more efficient than removing the last element in a linked list, in the worst case.
    - [✓ b. Appending an element to the end of an array, and appending an element to the end of a linked list, both cost $O(1)$, in the worst case.]

Two answers??

    - [✓ c. Switching between any two elements in an array, and switching between any two elements in a linked list, both cost $O(1)$, in the worst case.]
    - d. Prepending an element in the beginning of an array is less efficient than prepending an element in the beginning of a linked list.
    - e. none of the above.

6. Consider the following B-Tree of order 5.

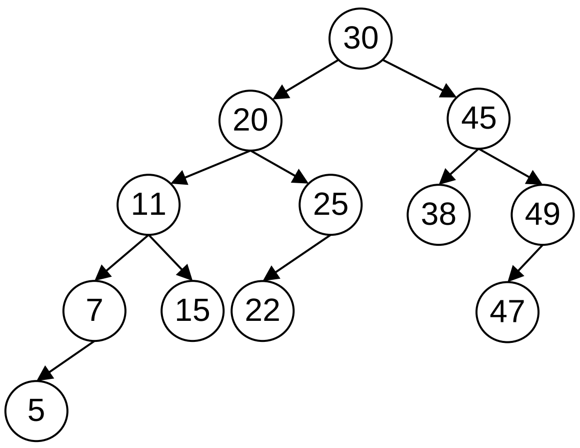


The resulting tree will definitely have one level less than the current tree after deleting the key
    - [✓ a. 45]
    - b. 27
    - c. 20
    - d. 18
    - e. all of the above.

7. The run-length encoding of the string CCCCBBBBAAAADDDD:
    - a. is CBAD4.
    - [✓ b. is C4B4A4D4.]
    - c. is 4CBAD.
    - d. can be both answers a and b.
    - e. can be all answers a, b and c.

8. The postfix expression: `9 9 7 – – 3 4 + 8 6 – * +` evaluates to
    - a. 0.
    - b. – 25.
    - c. – 7.
    - d. 17.
    - [✓ e. 21.]

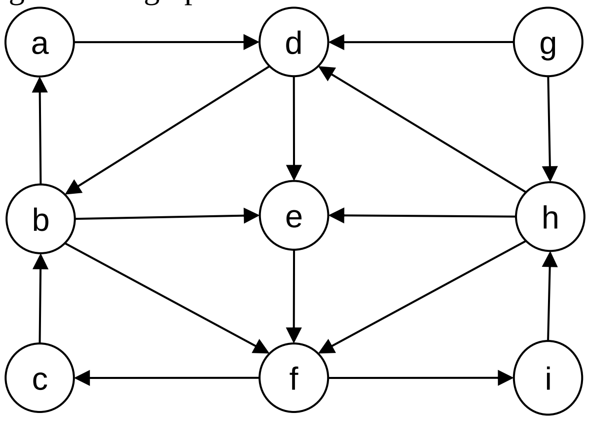
9. Consider the following AVL tree



The operation that **may** cause a single right rotation, without any double rotations on the AVL tree is:

a. inserting Key 48
b. deleting Key 38
c. deleting Key 45
d. deleting Key 5
e. none of the above

10. Consider the following directed graph



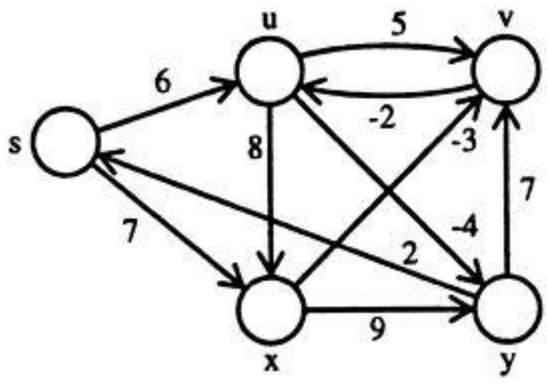
The number of strongly connected components is equal to

a. 5
b. 4
c. 3
d. 2
e. 1

{g},{a,b,c,d,e,f,h,i}

Q2. [20 points] (Dijkstra):

A. Consider the following weighted directed graph:



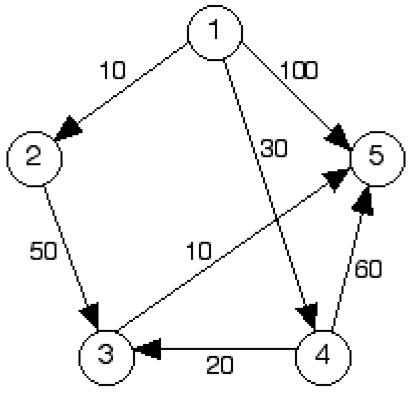
a) [4 points] Is it possible to solve the shortest path problem for the above graph? Justify your answer

Yes, since there are no negative cost cycles

b) [2 points] Is Dijkstra algorithm applicable on the above graph?

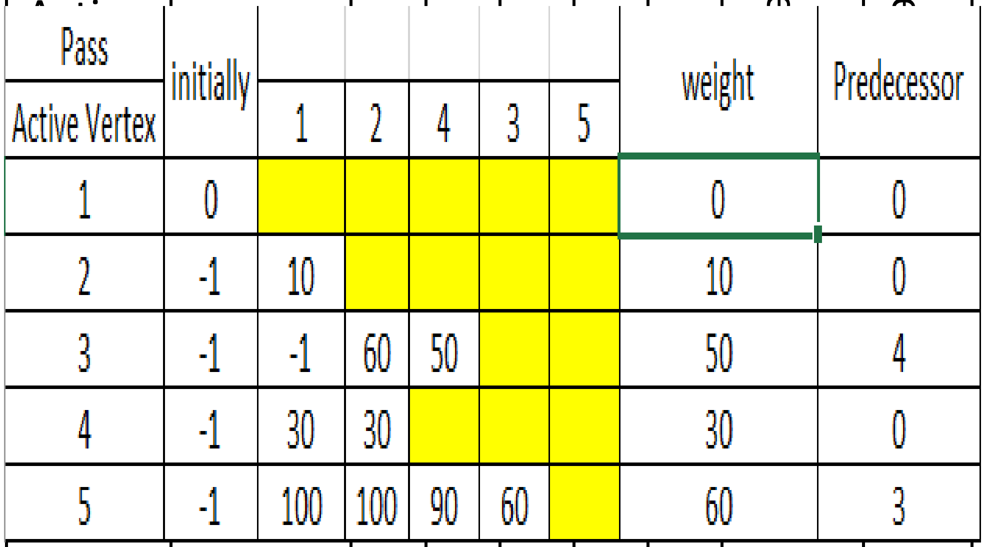
No, Djakarta algorithm only works for positive weights

B. Consider the following weighted directed graph:



a) [10 points] Apply the Dijkstra algorithm to find the shortest path to any vertex starting from vertex **1**.

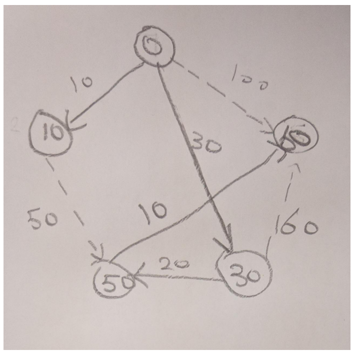
Pass	initially						we	Prede



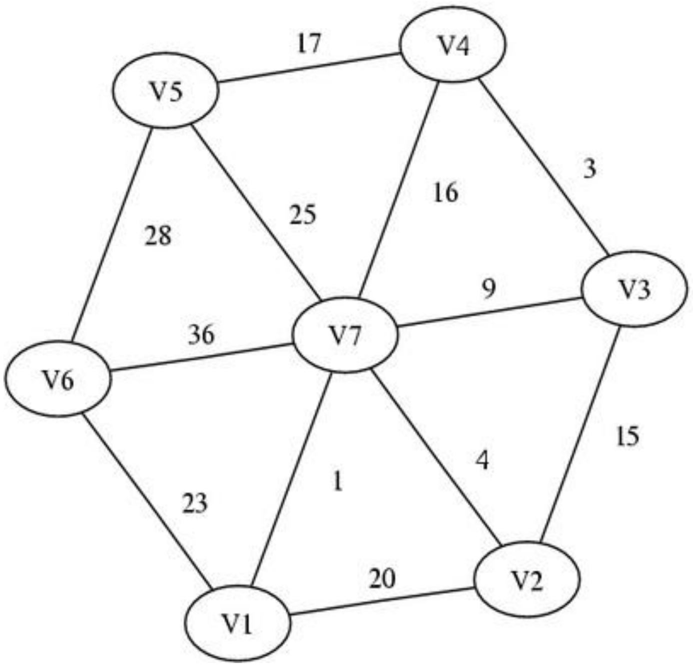
Pass	initially						weight	Predecessor
Active Vertex		1	2	4	3	5		
1	0						0	0
2	-1	10					10	0
3	-1	-1	60	50			50	4
4	-1	30	30				30	0
5	-1	100	100	90	60		60	3

5								

b) [4 points] Draw the resulting vertex-weighted graph.

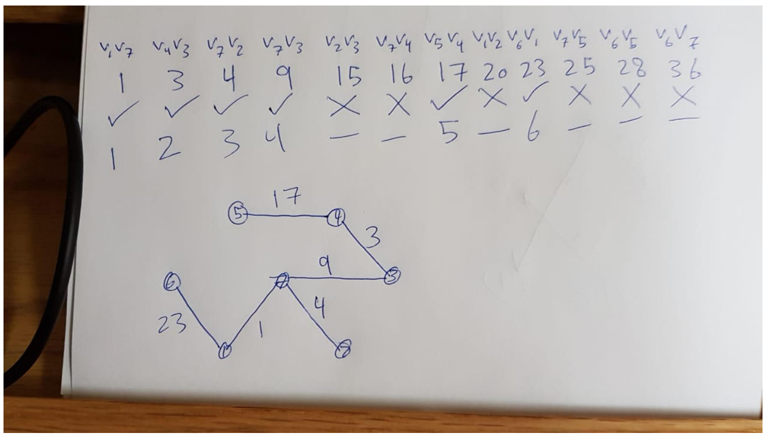


Q.3 [15 points] (Kruskal's)
Consider the following weighted undirected graph.



Apply Kruskal's algorithm to find a minimum spanning tree of the above graph.

edge												
weight												
Insertion status												
Insertion order												

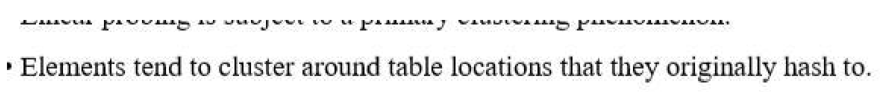


V_1V_7	V_4V_3	V_7V_2	V_7V_3	V_2V_3	V_7V_4	V_5V_4	V_1V_2	V_6V_1	V_7V_5	V_6V_5	V_6V_7
1	3	4	9	15	16	17	20	23	25	28	36
✓	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗
1	2	3	4	—	—	5	—	6	—	—	—

5
4
3
17
9
6
23
1
4

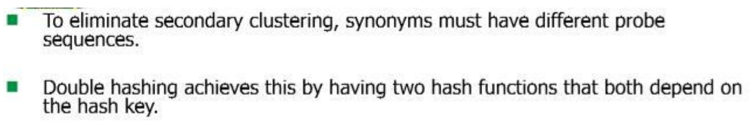
Q.4 [15 points] (Hashing):

a) [6 points] What is primary clustering? Clearly explain how double hashing avoids it.



Linear probing is subject to a primary clustering phenomenon.

- Elements tend to cluster around table locations that they originally hash to.



- To eliminate secondary clustering, synonyms must have different probe sequences.
- Double hashing achieves this by having two hash functions that both depend on the hash key.

b) [9 points] Consider inserting the following keys:

16 , 35 , 33 , 38 , 18 , 27 , 10

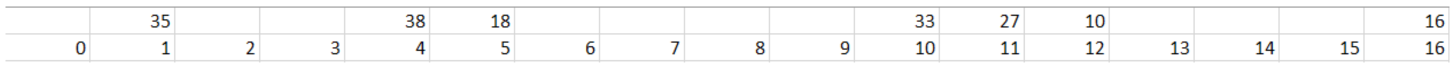
respectively, into a hash table of size 17, using open addressing and hash function:

$$h(key) = key \% 17$$

Use double hashing as a collision resolution policy with

$$h_p(key) = 11 - key\%11$$

Show the hash table after the insertions, showing all your work.

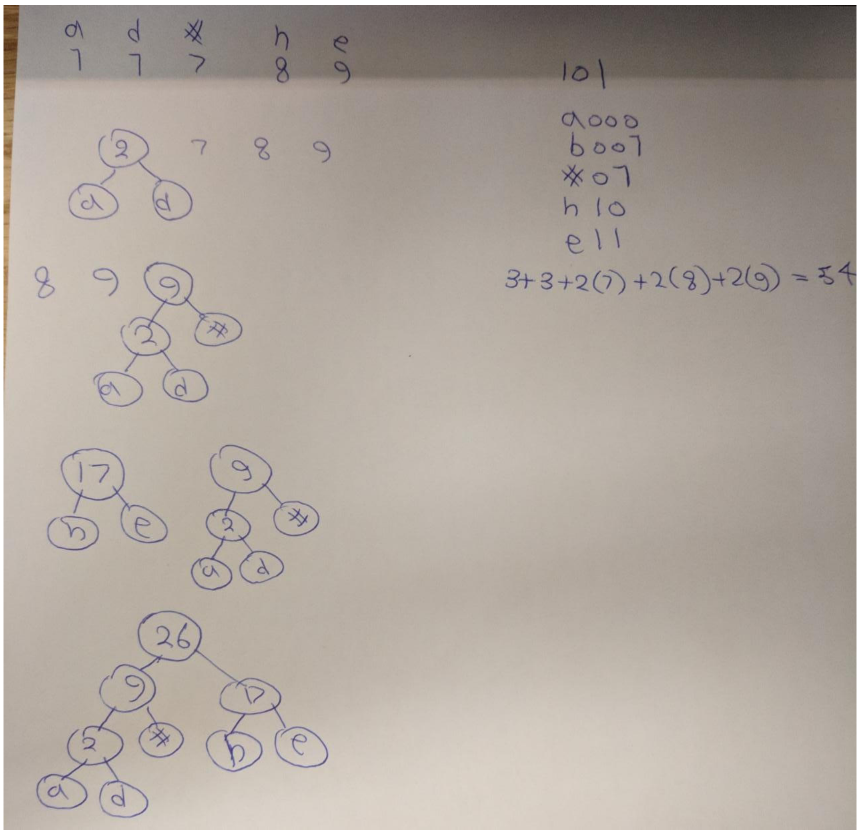


	35			38	18					33	27	10				16
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Q.5 [20 points]: (Compression)

a) [6 points] Using Huffman coding, show the resulting Huffman coding tree for compressing the following message. Make sure you show all your work.

h#he#hhee#he#hee#heed#hea#



a	d	#	h	e
1	1	7	8	9

101

a	000
b	001
#	01
h	10
e	11

$3+3+2(7)+2(8)+2(9)=54$

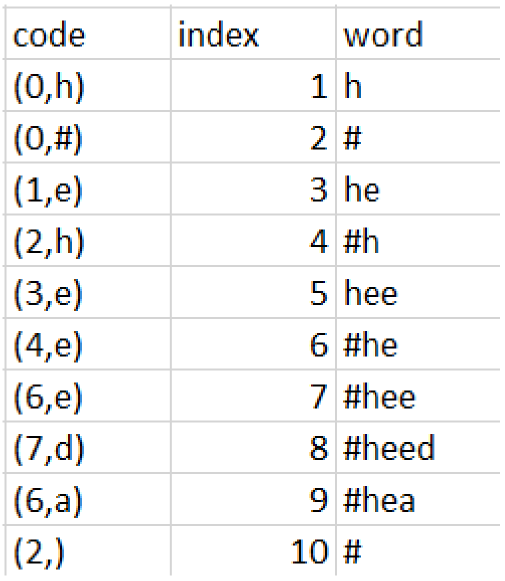
b) [4 points] Compute the compression ratio, showing your work. Make sure you state any assumptions.

Original message: 26*8=208
Compressed message: 3+5*8+54+12=109

109/208=52% of original data

c) [6 points] Compress the following message using LZ78. Make sure you show all your work:

h#he#hhee#he#hee#heed#hea#



code	index	word
(0,h)	1	h
(0,#)	2	#
(1,e)	3	he
(2,h)	4	#h
(3,e)	5	hee
(4,e)	6	#he
(6,e)	7	#hee
(7,d)	8	#heed
(6,a)	9	#hea
(2,)	10	#

d) [4 points] Compute the compression ratio, showing your work. Make sure you state any assumptions.

Original message: 26*8=208

Compressed message: 0h0#01e10h011e100e110e111d0110a0010
26+9*8=98

98/208=47% of original data

Quick Reference Sheet

```
public class SLLNode<T> {
    public T info;
    public SLLNode<T> next;
  public SLLNode();
  public SLLNode(T el)
  public SLLNode(T el, SLLNode<T> ptr);
}

public class SLL<T> {
    protected SLLNode<T> head, tail;
  public SLL();
  public boolean isEmpty();
  public void addToHead(T el);
  public void addToTail(T el);
  public T deleteFromHead();
  public T deleteFromTail();
  public void delete(T el);
  public void printAll();
  public boolean isInList(T el);
}

public class DLLNode<T> {
    public T info;
    public DLLNode<T> next, prev;
  public DLLNode();
  public DLLNode(T el);
  public DLLNode(T el, DLLNode<T> n,
                       DLLNode<T> p);
}

public class DLL<T> {
    private DLLNode<T> head, tail;
  public DLL();
  public boolean isEmpty();
  public void setToNull();
  public void addToHead(T el);
  public void addToTail(T el);
  public T deleteFromHead();
  public T deleteFromTail();
  public void delete(T el);
  public void printAll();
  public boolean isInList(T el);
}

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();
}

public class BSTNode<T extends Comparable<?
super T>> {
    protected T el;
    protected BSTNode<T> left, right;
    public BSTNode();
    public BSTNode(T el);
    public BSTNode(T el, BSTNode<T> lt,
                         BSTNode<T> rt);
}

public class BST<T extends Comparable<?
super T>> {
    protected BSTNode<T> root = null;
    public BST();
    protected void visit(BSTNode<T> p);
    protected T search(T el);
    public void breadthFirst();
    public void preorder();
    public void inorder();
    public void postorder();
    protected void inorder(BSTNode<T> p);
    protected void preorder(BSTNode<T> p);
    protected void postorder(BSTNode<T> p);
    public void deleteByCopying(T el);
    public void deleteByMerging(T el);
    public void iterativePreorder();
    public void iterativeInorder();
    public void iterativePostorder2();
    public void iterativePostorder();
    public void MorrisInorder();
    public void MorrisPreorder();
    public void MorrisPostorder();
    public void balance(T data[], int first,
                                  int last);
    public void balance(T data[]);
    public void insert(T el)
}
```

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2,$$

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}, \quad 2^{\lg n} = n$$