

King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department

First Semester 161 (2016/2017)

ICS 202 – Data Structures

Final Exam

Thursday, January 19th, 2017

Time: 120 minutes

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID# |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section 01 |  | Question # | Max Marks | Marks Obtained |
| Dr. Emad |  | 1 | 20 |  |
| 9-9:50am |  | 2 | 15 |  |
| Section 02 |  | 3 | 25 |  |
| Dr. Sami |  | 4 | 20 |  |
|  |  | 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 7 pages, including this page, plus one 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 [20 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<=5; k++)

sum++; // Statement 1

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

* 1. *n*2
  2. 5*n* log *n*
  3. 5 log2*n*
  4. 10*n* – 5
  5. none of the above.

1. Consider the following method
   1. public void final(int [] A, int n) {
   2. if (n ==0) {
   3. System.out.print(A[n]+" ");
   4. return;
   5. }
   6. final(A, n-1);
   7. System.out.print(A[n-1]+" ");
   8. final(A, n-1);
   9. }

The number of times the print statements in lines 3 and 7 are executed is equal to

* 1. 2*n* – 1
  2. log (*n*+1) + 1
  3. none of the above.

1. The method final in Question 2 when called on A=[5,4,3,2,1] and n = 2 outputs
   1. 4 3 4 5 5 5
   2. 5 4 5 3 5 4 5
   3. 0 0 0 1 0 0 0 3 0 0 0 1 0 0 0
   4. 5 5 5 4 5 5 5 3 5 5 5 4 5 5 5
   5. none of the above.
2. Consider the following B-Tree.



The resulting tree will definitely have one level less than the current tree after deleting the key

* 1. 27
  2. 18
  3. 20
  4. 45
  5. none of the above.

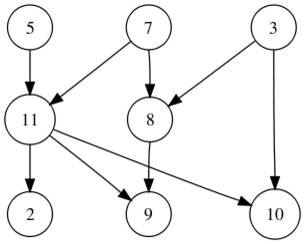
1. Consider the following directed graph

The number of strongly connected components is equal to

* 1. 1
  2. 2
  3. 3
  4. 4
  5. 5

**Q.2 [15 points] (Topological Sort):**

1. [3 points] Is Topological sort applicable on any type of graph? If yes explain why. If no mention the properties required to apply Topological sort.
2. [9 points] Show the steps of the topological sort algorithm on the following graph.



1. [3 points] Explain why the result of Topological sort may not be unique.

**Q.3 [25 points] (Graphs):**

Consider the following graph:

1

2

5

3

12

9

9

10

8

12

11

2

13

7

4

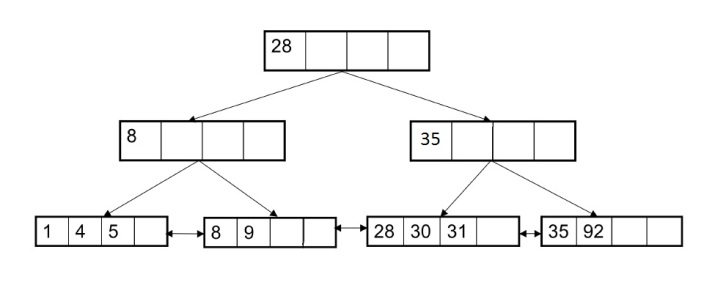
1

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pass** | **Initial distance** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **Edge Weight** | **Pred.** |
| **Active vertex** |  |  |  |  |  |  |  |  |  |  |  |
| **a** | 0 |  |  |  |  |  |  |  |  |  |  |  |
| **b** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **c** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **d** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **e** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **f** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **g** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **h** | ∞ |  |  |  |  |  |  |  |  |  |  |  |
| **i** | ∞ |  |  |  |  |  |  |  |  |  |  |  |

1. (12 points) Carry out Prim's algorithm starting from vertex **a** to find the minimum spanning tree of the above graph, by filling the above table.
2. (5 points) Carry out Kruskal's algorithm on the above graph. Show all your intermediate work.
3. (4 points) Assuming that the visitor method prints the label of the vertex in the graph, show the output of the post-order depth first traversal of the above graph.
4. (4 points) Assuming that the visitor method prints the label of the vertex in the graph, show the output of the pre-order depth first traversal of the above graph.

**Q.4 [20 points] (B+ Trees):**

Consider the following B+ Tree:



1. [2 points] According to the above representation, what are the values of M and L for the B+ Tree?
2. [3 points] When an internal node of the above B+ Tree becomes under-flow and when it becomes over-flow?
3. [3 points] When a leaf node of the above B+ Tree becomes under-flow and when it becomes over-flow?
4. [6 points] Show the above B+ Tree after inserting values 2 and 3.
5. [6 points] Show the above B+ Tree after deleting value 92.

**Q.5 [20 points]:** (Hashing and LZ-78 compression)

1. [10 points] Insert the following values in an open-addressing hash table of size 13 where the hashing function is K%13 and probing function is quadratic: i2 :

14, 11, 17, 12, 27, 1, 40, 24, 53, 25

1. [10 points] Compress the following message using LZ-78. (You must show the compression table):

BAABCAACCBAAABCA

**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 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)  } |

 ,  , ,  , , ,