CSC212-Mid 1 – Spring2017

CSC 212 Midterm 1 - Spring 2017

College of Computer and Information Sciences, King Saud University
Exam Duration: 90 Minutes

16/03/2017

Question 1 [30 points]

1. Choose the most appropriate answer:

(1) To show that $2n^3 \log n + 2n^3$ is $O(n^3 \log n)$, we can take $c = 4$ and n_0:

(a) -1 (b) -2 (c) 1 (d) O (e) 2

(2) Which of the following is **not** $O(n^2)$

(a) $n^2 \log n$ (b) $2n^2 + 3$ (c) $n(n+2)/2$ (d) n^2 (e) n

(3) Given an n-element array A of integers, an algorithm searches for the integer '9' and returns true if found. What is the **best**-case running of this algorithm.

(a) $O(\log n)$ (b) $O(n \log n)$ (c) $O(1)$ (d) $O(n^2)$ (e) $O(n)$

(4) We want to implement the method `retrieveAtIndex(int i)` which returns the i^{th} element of a list. Which representation would have the method running in $O(1)$?

(a) *LinkedList* (b) *ArrayList* (c) *DoubleLinkedList* (d) *a, b, and c* (e) *None*

(5) In the **worst** case, the method *remove* of the class *DoubleLinkedList* is :

(a) $O(1)$ (b) $O(n)$ (c) $O(\log n)$ (d) $O(n \log n)$ (e) $O(n^2)$

2. Consider the following code:

```
1 System.out.println("glhf");
2 for (int i = 0; i < n * log(n); i++)
3         for (int j = 2; j <= n; j++)
4                 System.out.println("op");
5 System.out.println("gg");
```

Choose the correct answer (**select an answer for each line**):

Line	Frequency				
1	(a) n	(b) -1	(c) 0	(d) 1	(e) $\log n$
2	(a) n	(b) n^2	(c) $n \log n + 1$	(d) $n \log n$	(e) $\log n$
3	(a) n^2	(b) $n^2 \log n$	(c) $n^2 + 1$	(d) $n(n \log n + 1)$	(e) $n(n+1)/2$
4	(a) $n - 1$	(b) n^3	(c) n^2	(d) $n(n \log n)$	(e) $(n-1)n \log n$
5	(a) 0	(b) n	(c) $n \log n$	(d) n^2	(e) 1
Total O	(a) $O(n^2 \log n)$	(b) $O(n^2)$	(c) $O(n \log n)$	(d) $O(1)$	(e) $O(n)$

Question 2 [35 points]

1. Write the method `public static <T> int lastIndex(List<T> l, T e)`, user of the ADT List, which returns the index of the last occurrence of e in l, or -1 if e does not exist. The first element has index 0.

   Example 2.1. *If $l : A, B, C, A, B, D$, then $lastIndex(l, "A")$ returns 3, $lastIndex(l, "C")$ returns 2 and $lastIndex(l, "F")$ returns -1.*

2. Write the method
   `public static <T> reverseCopy(DoubleLinkedList<T> l1, DoubleLinkedList<T> l2)`, user of *DoubleLinkedList*, which copies the elements of $l1$ to $l2$ in reverse order. The list $l1$ must not change. Assume that $l2$ is empty.

   Example 2.2. *If $l1 : A, B, C, D$, then calling* reverseCopy(l1, l2) *results in $l2 : D, C, B, A$.*

Question 3 [35 points]

1. Implement the method `public void cut(int k)`, member of the class *DoubleLinkedList*, which removes the last k elements of the list. The method moves *current* to the first element if the resulting list is not empty. Assume that $0 < k \leq$ the length of the list. Do not call any method of the class *DoubleLinkedList* and do not use any extra data structures. The method *cut* must be $O(n)$.

   Example 3.1. *If $l : A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow E \leftrightarrow F$, then calling* l.cut(3) *results in $l : A \leftrightarrow B \leftrightarrow C$. After calling* l.cut(6), *l becomes empty.*

2. Implement the method `public void remove(T e)`, member of the class *LinkedList*, which removes all occurences of e. The method moves *current* to the first element if the resulting list is not empty. Do not call any method of the class *LinkedList* and do not use any extra data structures. The method *remove* must be $O(n)$.

   Example 3.2. *If $l : A \rightarrow B \rightarrow C \rightarrow B \rightarrow E \rightarrow B$, then calling* l.remove("B") *results in $l : A \rightarrow C \rightarrow E$.*

CSC212-Mid 2 – Spring2017

CSC 212 Midterm 2 - Spring 2017

College of Computer and Information Sciences, King Saud University
Exam Duration: 90 Minutes

23/04/2017

Question 1 [35 points]

1. Write the method `public static <T> int search(Stack<T> st, T e)` (user of the ADT Stack) that returns the position of the first occurrence of e in st (the top element is at position 1). Use the method **equals** to test for equality. If e does not exist the method returns -1. The stack st must **remain unchanged** after the call.

   Example 1.1. *If* $st : B, E, B, C, B, A$ *(from top to bottom), then calling* search(st,"B") *results in* 1, *calling* search(st,"C") *results in* 4 *and calling* search(st,"G") *results in* -1.

2. Trace the evaluation of the following expression (draw the stack after every push):
   `5 1 2 + 4 * + 3 -`

3. Trace the evaluation of the following expression (draw the stacks after every push):
   `8 / 2 < 2 + 5 - 3`

Question 2 [30 points]

1. Complete the linked implementation of the ADT Queue below. Write the methods: *length, full, enqueue* and *serve*.

```
public class LinkedQueue<T> implements Queue<T> {
        private Node<T> head, tail;
        private in size = 0;
        public LinkedQueue() {
                head = tail = null;
                size = 0
        }
}
```

   Add the method `public T serveTail()`, which returns and removes the last element in the queue.

2. Write a recursive method `private int countLeafs(BTNode<T> t)`, member of the class *BT*, which returns the number of leaf nodes in the sub-tree *t*. (Do not call any methods of the class BT. Do not define any data members. Non-recursive methods are not accepted).

   Example 2.1. *For the binary tree shown in Figure 1,* `countLeafs` *returns 4. For the one shown in 2, the method returns 2.*

Question 3 [35 points]

1. Write the **preorder, inorder, and postorder** traversal of the elements in tree shown in Figure 1.

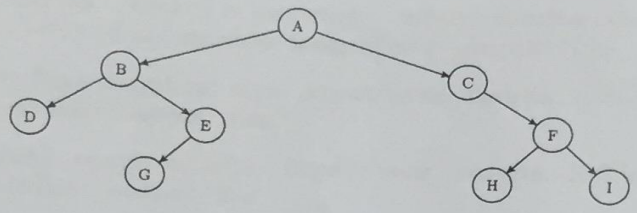


Figure 1: Binary Tree.

2. Consider the following Binary Tree:

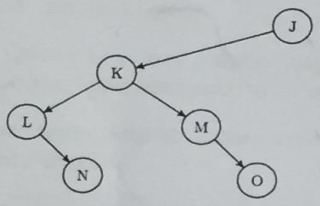


Figure 2: Binary Tree.

Given the initial Binary Tree shown in **Figure 2**, draw the resulting tree after each of the following sequences of operations. **For each sequence, you should draw one final tree result. Each sequence should be applied on the original tree.**

(a) find(Root); insert('P', RightChild); insert('Q', RightChild); find(Parent); insert('R', LeftChild)

(b) find(Root); find(LeftChild); find(LeftChild); DeleteSub( ); find(LeftChild); insert('P', LeftChild)

(c) find(Root); find(RightChild); insert('P', RightChild); find(Root); find(LeftChild), DeleteSub( )

3. Starting with an **empty tree**, draw the resulting Binary Tree after each of the following sequences of operations **For each sequence, you should draw one final tree result. Each sequence should be applied on an empty tree.**

(a) insert('A', Root); insert('B', LeftChild); insert('C', RightChild); find(Root); insert('D', RightChild);

(b) insert('A', Root); insert('B', RightChild); insert('C', LeftChild); insert('D', RightChild); find(Root); insert('E', LeftChild);

(c) insert('A', LeftChild); insert('B', Root); insert('C', Parent); insert('D', LeftChild);

CSC 212 - Spring 2017

College of Computer and Information Sciences, King Saud University
Exam Duration: 3 Hours

13/05/2017

(1) All answers must be written on the answer sheet. (2) Calculators are not allowed.

Question 1 [16 points]

1. Consider the following code:

```
1 for (int i = 0; i < n; i++) {
2         System.out.println("first");
3         for (int j = 0; j <i; j++)
4                 System.out.println("second"); }
5 System.out.println("good bye");
```

Choose the correct answer:

Line			Frequency		
1	(a) n	(b) $n+1$	(c) n^2	(d) 0	(e) $n+2$
2	(a) n	(b) $n+1$	(c) n^2	(d) 0	(e) $n-1$
3	(a) n	(b) n^2	(c) $n \log n$	(d) 1	(e) $n(n+1)/2$
4	(a) n	(b) n^2	(c) $n(n-1)/2$	(d) 1	(e) 1
5	(a) n	(b) n^2	(c) 0	(d) 1	(e) $n \log n$
Total O	(a) $O(n)$	(b) $O(n^2)$	(c) $O(n^3)$	(d) $O(n*i)$	(e) $O(1)$

2. Consider the following code:

```
1 int sum = 0;
2 for (int i = 0; i <= n; i++)
3   for (int j = 2; j <n-1; j++)
4         sum += i;
5 return sum;
```

Choose the correct answer:

Line			Frequency		
1	(a) n	(b) 1	(c) n^2	(d) 0	(e) i
2	(a) n	(b) $n+1$	(c) $n+2$	(d) $n-1$	(e) n^2
3	(a) $(n+1)(n-3)$	(b) $n(n-2)$	(c) $(n+1)(n-2)$	(d) $n^2(n+1)$	(e) $n(n+1)$
4	(a) $(n+1)(n-2)$	(b) $n(n-2)$	(c) $(n+1)(n-1)$	(d) $n^2(n+1)$	(e) $n(n+1)$
5	(a) n	(b) n^3	(c) n^2	(d) 1	(e) n
Total O	(a) $O(n)$	(b) $O(n^2)$	(c) $O(n^3)$	(d) $O(n^4)$	(e) $O(1)$

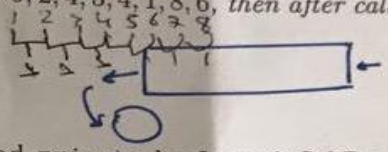
Question 2 [16 points]

1. Write a method `static Stack <T> removeFirst(Stack<T> s, T e)`, user of Stack ADT. It accepts a stack s and an element e. It creates and returns a new stack with all elements except the first instance of e from the top if exists. The stack s becomes empty at the end. Use the method equals to test for equality.

   Example 2.1. *Suppose s contains A, O, M, K, O, A (from top to bottom). When calling* removeFirst (s, O), *it will return A, M, K, O, A and when calling* removeFirst (s, B), *it will return A, O, M, K, O, A.*

2. As a user of ADT Queue, write the method `public static void bubble(Queue<Integer> q)`, which compares consecutive items and exchanges those that are out of order (assume the order must be decreasing). **Do not use any extra data structures.**

   Example 2.2. *If q contains* 5, 2, 4, 3, 4, 1, 8, 6, *then after calling the method* bubble, *q should become* 5, 4, 3, 4, 2, 8, 6, 1.



Question 3 [16 points]

1. Write the **recursive** method `private boolean inSubTree(BTNode<T> t, T e)`, member of the class *BT* which returns true if the data *e* exists in the subtree *t*, false otherwise.

2. Write the method `public boolean insertUnique(T e)` (member of *LinkedList*), that inserts *e* if it is not already in the list. If *e* does not exist, the method behaves in the same way as *insert*. The method returns true if *e* is inserted, false otherwise. **Do not call any methods of the class LinkedList.**

   Example 3.1. *If the list l contains A, B, C, D and* current *is on C, then calling* l.insertUnique("A") *does not change the list. Calling* l.insertUnique("E") *results in A, B, C, E, D with* current *on E.*

Question 4 [12 points]

1. Consider the following heap represented as an array: **3, 5, 6, 6, 8, 9**. Choose the correct answer for every operation (all operations are done on the above heap).

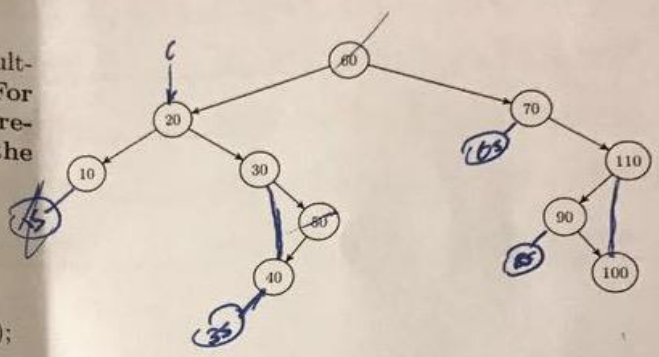
1. Heap after inserting 3:				
(a) 3,5,3,6,8,6,9	(b) 3,5,6,6,8,9,3	(c) 3,3,5,6,8,9,6	(d) 3,5,3,6,8,9,6	(e) 3,5,3,9,8,6,6
2. Heap after inserting 7:				
(a) 7,5,6,6,8,9,3	(b) 3,5,7,6,8,9,6	(c) 3,5,6,6,8,9,7	(d) 3,6,5,6,8,9,7	(e) 3,5,7,6,8,9,6
3. Heap after inserting 1:				
(a) 6,5,3,6,8,9,1	(b) 3,5,1,6,8,9,6	(c) 1,3,5,6,8,9,6	(d) 3,5,6,6,8,9,1	(e) 1,5,3,6,8,9,6
4. Heap after inserting 5:				
(a) 5,5,3,6,8,9,6	(b) 3,5,5,6,8,9,6	(c) 3,5,6,6,8,9,5	(d) 3,5,5,6,8,6,9	(e) 3,5,5,6,8,6,9
5. Heap after inserting 2:				
(a) 2,5,3,6,8,9,6	(b) 3,5,2,6,8,9,6	(c) 3,5,6,6,8,9,2	(d) 2,3,5,6,8,9,6	(e) 2,5,6,6,8,9,3

2. Consider the following heap represented as an array: **9, 8, 7, 8, 6, 5, 4**. Choose the correct answer for every operation (all operations are done on the above heap).

1. Heap after deleting one key:				
(a) 8,7,8,4,6,5	(b) 4,8,7,8,6,5	(c) 9,8,7,8,6,5	(d) 8,4,7,8,6,5	(e) 8,8,7,4,6,5
2. Heap after deleting two keys:				
(a) 8,6,7,4,5	(b) 8,7,4,6,5	(c) 8,8,7,4,6	(d) 8,6,7,5,4	(e) 7,8,5,4,6
3. Heap after deleting three keys:				
(a) 6,7,4,5	(b) 7,6,5,4	(c) 8,6,7,4	(d) 5,6,7,4	(e) 7,5,6,4

Question 5 [12 points]

Given the initial BST to the right, draw the resulting BST for each of the following sequences. **For each part, you should have one final tree result. Each part should be applied on the original tree.**



1. insert(35); insert(65); insert(85); insert(15);
2. insert(15); removeKey(50); findKey(20); insert(19);
3. removeKey(60); removeKey(70); removeKey(20); removeKey(10);
4. removeKey(70); removeKey(60); removeKey(110); removeKey(90);

Question 6 [12 points]

1. Trace the evaluation of the following postx expression using a stack. Draw the stack after every push or pop operation (you have to draw the stack 13 times in total):
   `9 8 - 7 8 - / 9 2 + 7 + *`
2. Trace the evaluation of the following expression (draw the stacks after every push operation):
   $7 + 8 - 2 \leq 2 * 5 + 3/2$

Question 7 [12 points]

Use the hash function $H(key) = key\%11$ to store the sequence of keys **24,27,19,13,35,16,30,38,57,8** in the hash table. Use the following collision resolution strategies:

1. Linear rehashing (c=1), indicate the number of probes.
2. External chaining.
3. Coalesced chaining with cellar size 3 (do not change the hash function).

Question 8 [4 points]

Choose the **most appropriate** data structure:

1) Check that the parentheses in an expression are balanced
(a) Hash table (b) array list (c) linked priority queue (d) heap (e) linked stack

2) Evaluate a postfix expression
(a) linked list (b) linked queue (c) binary tree (d) AVL tree (e) array stack

3) Implement a phone directory
(a) AVL tree (b) array list (c) heap (d) Hash table (e) linked list

4) Undo/redo in a word processor
(a) linked stack (b) array list (c) linked priority queue (d) heap (e) linked list

5) Evaluate an infix expression
(a) linked list (b) linked queue (c) binary tree (d) linked stack (e) array queue

6) Manage clients' orders at an online store
(a) linked stack (b) heap (c) array queue (d) priority queue (e) Hash table

7) Check that HTML tags are balanced
(a) linked list (b) linked queue (c) binary tree (d) linked stack (e) array queue

8) Manage patients in an emergency service
(a) linked stack (b) heap (c) array queue (d) priority queue (e) Hash table

ADT Queue Specification

- enqueue (Type e): **requires**: Queue Q is not full. **input**: Type e. **results**: Element e is added to the queue at its tail. **output**: none.
- serve (Type e): **requires**: Queue Q is not empty. **input**: none. **results**: the element at the head of Q is removed and its value assigned to e. **output**: Type e.
- length (int length): **requires**: none. **input**: none. **results**: The number of elements in the Queue Q is returned. **output**: length.
- full (boolean flag): **requires**: none. **input**: none. **results**: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.

ADT Stack Specification

- Push (Type e): **requires**: Stack S is not full. **input**: Type e. **results**: Element e is added to the stack as its most recently added elements. **output**: none.
- Pop (Type e): **requires**: Stack S is not empty. **input**: **results**: the most recently arrived element in S is removed and its value assigned to e. **output**: Type e.
- Empty (boolean flag): **requires**: none. **input**: none. **results**: If Stack S is empty then flag is true, otherwise false. **output**: flag.
- Full (boolean flag): **requires**: none. **input**: none. **results**: If S is full then Full is true, otherwise Full is false. **output**: flag.

CSC212-Quiz 1 – Spring2017

Question1 [15 points]

For the following method, write down the S/E, frequency and total for every line, the total number of steps of the method and its O notation

```
1: public void func(int n){
2:         System.out.println("hello world");
3:         for ( int i = 0 ; i < n ; i ++ ) {
4:                 for ( int j = n ; j > 2 ; j--)
5:                         System.out.println("test"); }
6:         System.out.println("bye"); }
```


Question2 [15 points]

As a user, write the method *replace* that replaces all the occurrences of the element e in list l with the element *r*. Do not use any auxiliary data structure. The method signature is public static<T> void *replace (List<T> l, T e, T r)*.

Example 1.1. *For example, if the List l is A; B; A; D; E; D, then replace(l, A, F) makes the list F; B; F; D; E; D*

CSC212-Mid 1(Makeup) – Spring2017

CSC 212 Midterm 1 Makeup Exam - Spring 201

College of Computer and Information Sciences, King Saud Universi
Exam Duration: 90 Minutes

03/05/2017

Question 1 [30 points]

1. Choose the most appropriate answer:

(1) The function $2n \log n + n \log(n^{n^2}) + 3n^2$ is:

(a) $O(n^2 \log n)$ (b) $O(n \log n)$ (c) $O(n^2)$ (d) $O(n \log(n^n))$ (e) $O(n^$

(2) Which of the following is **not** $O(n)$

(a) $\log(2^n + 1)$ (b) $\sum_{i=0}^{n-1} 1$ (c) $\log(3^{n-1})$ (d) $\sum_{i=1}^{n} \log(n+i)$ (e) log(

(3) Given an n-element array A of positive integers, an algorithm selects log
at random from A and executes for each element $A[i]$ of these elements $O($
What is the **best**-case running of this algorithm?

(a) $O(\log n)$ (b) $O(n \log n)$ (c) $O(1)$ (d) $O(n^2)$ (e) $O(n$

(4) We want to implement the method `removeLast()` which removes the
from the data structure. Which implementation would have the metho
$O(1)$?

(a) *ArrayQueue* (b) *ArrayList* (c) *LinkedQueue* (d) *a and b* (

(5) In the **best** case, the method *remove* of the class *ArrayList* is :

(a) $O(\log n)$ (b) $O(n)$ (c) $O(n^2)$ (d) $O(1)$ (e) O

2. Consider the following code:

```
1 int sum = 0;
2 for (int i = 1; i <= n; i++)
3   for (j = 0; j <= 2*n; j+=2)
4     sum = i + j;
5 System.out.println(sum);
```

Frequency				
) 1	(b) -1	(c) 0	(d) n	(e) 2
a) n	(b) $n-1$	(c) $n+1$	(d) $n+2$	(e) $n-2$
(a) $(n+1)(2n+1)$	(b) $n(2n+1)$	(c) $n(n+1)$	(d) $n(n+2)$	(e) $n(n+1)/2$
(a) $n(n+1)$	(b) $2n^2$	(c) $2n(n+1)$	(d) n^2	(e) $n^2/2$
(a) n	(b) -1	(c) 0	(d) 1	(e) 2
(a) $O(n^3)$	(b) $O(n^2)$	(c) $O(n \log n)$	(d) $O(1)$	(e) $O(n)$

stion 2 [35 points]

Write the method `public static <T> void commonE(List<T> l1,List<T> l2,List<T>` user of the ADT List, which inserts the common elements between list *l1* and list *l2* in the list cl. Assume that the element in *l1* and *l2* are unique and the List c is initially empty.

Example 2.1. *If* l1 : A, B, C, F, M, D, *and* l2 : R, M, W, F, *calling* commonE(l1, l2, cl) *results in* cl : F, M

2. Write the method
`static <T> boolean revSublist(DoubleLinkedList<T> l1, DoubleLinkedList<T> l2)`
user of the ADT *DoubleLinkedList* which returns true if *l1* appears as a contiguous sublist in reverse order in *l2*, false otherwise. Assume that *l1* is not empty.

Example 2.2. *Given* l2 : A, B, C, D, E, F, G, *then the method returns true for* l1 : E, D, C, *false for* l1 : G, F, D, *true for* l1 : C *and false for* l1 : A, B, C,

Question 3 [35 points]

1. Implement the method `public void remove(T[] elem,int n)`, member of the class *LinkedL* which removes all the elements of *elem* from the list. If the element pointed by *current* is removed, *current* is moved to the first element of the list. The array *elem* has n elements. Assume that the list is not empty. Do not call any method of the class *LinkedList* and do not use any extra data structures.

Example 3.1. *If* $l : A, B, C, F, C, K, D$, *then calling* l.remove([C,D,F],3) *results in then* $l : A, B, K$

2. Write the method `public void insertInv()`, member of the class *DoubleLinkedList*, whicl inserts the inverse of the double linked list after the last element without changing th position of the *current*. Assume that the double linked list is not empty. Do not use an extra data structures and do not call any method of the class.

Example 3.2. *If* $l : A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow E \leftrightarrow F$, *then calling* l.insertInv() *results* $l : A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow E \leftrightarrow F \leftrightarrow F \leftrightarrow E \leftrightarrow D \leftrightarrow C \leftrightarrow B \leftrightarrow A$.