15-111 Introductions to Data Structures

Summer II - 09

Midterm Exam – 80 minutes

Instructions: You are allowed to bring one page of notes. You will have access to APIs'. No other material allowed. Please write your answers clearly and legibly to receive full credit.

Name: ______Key______ (please print)

2 pts Each

1. Find the best answer for the following multiple choice questions. (20 points)
   a. Java is
      i. Procedural [✓ ii. Object oriented] iii. Functional iv. None of the above
   b. A Java interface is
      i. Extended by a class
      [✓ ii. Implemented by a class]
      iii. Contains methods that are already implemented
      iv. None of the above
   c. The runtime of an algorithm depends on
      i. Data size
      ii. Processor speed
      iii. Language
      iv. RAM
      [✓ v. All of the above]
      vi. None of the above
   d. Inserting an element to the beginning of an array (that is A[0] element) is more difficult than inserting an element to the beginning of a linked list.
      [✓ i. TRUE]
      ii. FALSE
   e. Suppose Node is a class that contains two integer fields and a pointer to next Node. Consider the following definitions Node M = new Node(); Node N = M; The number of bytes required to hold M and N are
      i. M = 4 bytes, N = 8 bytes
      ii. M = 8 bytes, N = 4 bytes
      [✓ iii. M = 12 bytes, N = 4 bytes]
      iv. M = 4 bytes, N = 12 bytes
      v. None of the above

f. Given a collection of algorithms that runs on O(1), O(n log n), O(n), $O(n^2)$, O(log n), O(n!), order the algorithms from fastest to slowest

   i. O(1), O(n log n), O(n), $O(n^2)$, O(log n), O(n!)

   [✓ ii. O(1), O(log n), O(n), O(n log n), $O(n^2)$, O(n!)]

   iii. O(1), O(log n), O(n log n), O(n), $O(n^2)$, O(n!)

   iv. None of the above

g. Suppose that the complexity of an algorithm is $O(n^2)$. Suppose that the program that uses the algorithm run in 10 seconds for a data set of size n. If the data size is doubled, how long will it take (approximately) to run the program?

   i. 10 seconds

   ii. 100 seconds

   iii. 6-7 minutes

   [✓ iv. None of the above]

h. What is the best data structure to solve the following problem? A list needs to be built dynamically. Data must be easy to find, preferably in O(1). The user does not care about any order statistics such as finding max or min or median.

   i. Use an Array given index

   ii. Use a Singly LL

   iii. Use a Stack

   iv. Use a Queue

   v. None of the above → not given index

i. Finding the max element in an unordered stack would require

   i. O(1) operations

   ii. O(log n) operations

   [✓ iii. O(n) operations]

   iv. None of the above

j. Suppose that recursive function foo(n) calls itself twice within its body(assume foo(n-1) called twice). Also assume the recursion would end when n=0. How many function calls does foo initiates?

   i. O(n)

   ii. $O(n^2)$

   [✓ iii. $O(2^n)$]

   iv. O(n!)

   v. None of the above

2. [10 points] Complexity of Algorithms. For each of the following Algorithms, find the complexity of the algorithm using big O notation. You must justify your answer with 1-2 lines of explanation. Choose from O(1), O(n log n), O(n), O(n^2), O(log n), O(n!). Note that you can ignore any constant factors in the expression, That is O(4n) is equivalent to O(n)

a. Determining all duplicates in a singly LL.

$O(n^2)$ or if sorted $O(n \log n)$

b. An algorithm that determines all possible paths between n cities.

$O(n!)$

2.5 pts each

For each of the following, count the number of operations where some_statement is executed

based on the loops

c. for (int I = 0; I < n; I +=2) → $O(n/2)$

for (int j = 1; j < n; j++) $O(n)$

{some_statement;} → $\boxed{O(n^2)}$

d. for (int j = 1 ; j < n ; j *= 2) → $O(\log n)$

for (int I = 1; i<n; i++) → $O(n)$

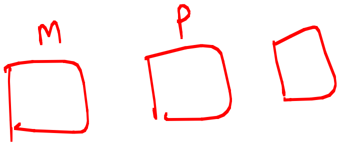
{some_statement;} → $\boxed{O(n \log n)}$

3. [15 points] Assume that LL is a DOUBLY linked list with the head node and at least one other internal node M which is not the last node. Write few lines of code to accomplish the following. You may assume that each node has a **next** pointer and **prev** pointer. You may NOT swap data to accomplish any of the following operations. For each operation, assume the **original list** as described above. You are encouraged to draw pictures to justify your code. Note that for each operation, you need to manipulate at least two pointers, next and prev.

5 pts a. Delete the head node

```
head = head.next;
head.prev = null;
```

partial Credit OK

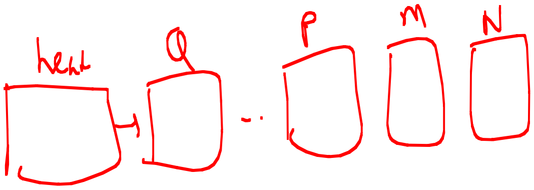


5 pts b. Insert a node P immediately after M

```
P.next = M.next;
M.next = P;
(P.next).prev = P;      P.prev = M;
```

10 pts c. Swap head and the node M (you may not swap data)

```
Q = head.next
P = m.prev;
N = m.next;
```



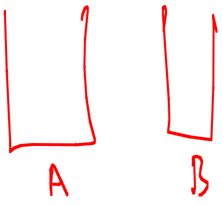
```
P.next = head;          m.next = Q
head.prev = P;          Q.prev = m
head.next = N;          M.prev = null;
N.prev = head           head = m;
```

4. [10 points] Complete the following method in the LinkedList class. The method contains is supposed to return true if the there is a node in the list that is equal to the given Comparable c. You can assume the Node class has the public fields, data (a Comparable) and next (a pointer to another Node)

```
public boolean contains(Comparable c) {
      Node ptr = head;
      While ((ptr != null) && (ptr.data).CompareTo(c) != 0)
                 ptr = ptr.next;
      if (ptr == null) return False;
       else return true;
}
```

-2 pts if they miss this

5. [10 points] Suppose you were asked to write a method that will take two sorted stacks A and B (min on top) and create one stack that is sorted (min on top). You are allowed to use only the stack operations such as pop, push, size and top. No other data structure such as arrays are not allowed. You are allowed to use stacks. Note that elements on the stack can be compared using compareTo.



```
Public  Stack  mergeSortedStacks(Stack A, Stack B) {
    Stack C = new Stack();
    While (!A.empty() && !B.empty()) {
           if ((A.top()).CompareTo(B.top()) < 0)
                   C.push(A.pop());
           else
                   C.push(B.pop());
    if (A.empty())  while (!B.empty())  C.push(B.pop());
    if (B.empty())  while (!A.empty())  C.push(A.pop());
```

There could be many soln's use discretion partial credit ok

```
Stack D = new Stack();
while (!C.empty()) D.push(C.pop()); return D;
```

5

6. [10 pts] Consider the following recursive method.

```
public  int foo(int a, int b)
{
        if  (a%b == 0)
          return b;
      else
         return foo(b, a%b);


}
```

3 a. What is the output given by foo(17, 3)?

$foo(17,3) \rightarrow foo(3,2) \rightarrow foo(2,1) = \boxed{1}$

3 b. What is the output given by foo(3, 9) ?

$foo(3,9) \rightarrow foo(9,3) \rightarrow \boxed{3}$

4 c. Explain briefly the purpose of the foo function. (example: it is finding the max of x and y)

GCD(a,b)

7. [10 pts] Given a Queue Q, write a method that will find the max element in the queue. You may only use queue operations such as eneque, dequeue, size etc.. No other data structure can be used other than queues. Queue must remain intact after finding the max. The elements in the queue can be compared using compareTo method

```
public  Comparable  findMax(Queue Q) {
    Comparable max = Q.deque();
    Q.enque(max);
    for (int i = 1; i < Q.size(); i++) {
        Comparable nxt = Q.deque();
        if (nxt.compareTo(max) > 0)  max = nxt;
        Q.enque(nxt);
```

3

There could be many soln
use discretion
partial credit ok

```
        return max;
    }
```

8. [5 pts] Write a method findLast that will find the last node of a LL. Return a pointer (reference) to the last node

```
public  Node  findLast(LinkedList  myLL){

        Node ptr = myLL.head
        While (ptr != null && ptr.next != null)
              ptr = ptr.next;

        if (ptr == null) return null;
          return ptr
}
```

-2 pts if miss this

9. [10 pts] For each of the following scenarios choose the “best” data structure from the following list or a **combination** of data structures: an unsorted array, linked list, DLL, circular LL, stack, queue. In each case, justify your answer briefly.

a. Suppose that a grocery store decided that customers who come first will be served first

Queue

b. A list must be maintained so that any element can be accessed randomly

2 each

Array

c. A program needs to remember operations it performed in opposite order

Stack

d. The size of a file is unknown. The entries need to be entered as they come in. Entries must be deleted when they are no longer needed. It is important that structure has flexible memory management

LL

e. A list must be maintained so that elements can be added to the beginning or end in O(1)

Circular DLL

Extra Credit: (10 pts) write a method **createArrayOfLL** that takes a file of words, one in each line and creates an array of linked list nodes as follows.

(a) Create an array of 26 linked lists
(b) Open the file and Insert each word into the array based on its first letter, eg: any word that begins with ‘a’ goes to A[0] list etc..
(c) Return the reference to the array of linked lists

You must consider ALL cases . You must write reasonable comments to describe your algorithm. You may assume public fields for each node, data and **next**. There WILL NOT be any partial credit for this problem. The question will be graded ALL or NOTHING. [Use extra paper, if necessary to write the answer]. Node class is defined as follows.

```
Public Node[] createArrayOfLL(String inputfile) {
```