King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering
Information and Computer Science Department
Second Semester 172 (2017/2018)

ICS 202 – Data Structures
Major Exam 1
Saturday, February 24th, 2018
Time: 90 minutes

Name: ____

ID#									

Section 01 Dr. Wasfi 8-8:50am
Section 02 Dr. Emad 09-09:50am

Question #	Max Marks	Marks Obtained
1	20	
2	20	
3	20	
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 9 pages, including this page and the 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 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. You are allowed to use the methods in the reference sheet, unless instructed otherwise in the question.
11. 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) Answer the following questions:

1. (5 points) Suppose that you have to choose a data structure of your program between:
   (a) SLL
   (b) DLL

Which criteria you need to consider in order to decide about which structure to choose?

DOUBLY LINKED LIST DLL has both a previous and a next pointer. Hence any node of the linked list contains address of both the previous and the next node. This is very useful when you need to traverse both ways in a linked list and deletion of specific nodes.

SINGLE LINKED LIST SLL only has a next pointer. Hence a node can reference only the next node in the linked list. SLL is simpler in terms of implementation, and typically has a smaller memory requirement, as it only needs to keep the forward member referencing in place. But some operations are expensive such as deleting from tail.

2. (5 points) Which one of the following data structures is more suitable to implement a Queue:
   (a) Fixed size array
   (b) Single Linked List
   (c) Doubly Linked List

Justify your answer:

SLL, SLL is simpler in terms of implementation, and typically has a smaller memory requirement. Queue only needs one ends to add (tail), and another end to delete (head). No need for expensive operations such as deleting from tail in queue implementation.

3. (5 points) Mention and explain the feature in objected-oriented languages that enables software reuse.
   Inheritance. Through inheritance, one can reuse in the subclass all the methods that have been implemented in the super class(es).

4. (5 points) Assume that a priority queue is to be implemented using a doubly linked list, where order is maintained by putting a new element in its proper position according to its priority. What is the time complexity of the ***`enqueue(el)`*** and ***`dequeue()`*** operations, given a queue with n elements? Briefly justify your answer.
   `enqueue(el)` will cost $O(n)$ time, since in the worst case, we need to search for the right place where to insert the element `el`.
   dequeue() will cost O(1), since dequeuing removes the first/highest priority element, which always exists at the beginning of the queue.

Q.2: (20 points) Complexity Analysis

(a) (10 points) Given that ***n*** is the problem size, count the number of times **MyStatement** gets executed. Then, express the cost of the piece of code in terms of big *O*() notation.

```
for (i=1; i <= √n ; i++) {
   sum[i] = 0;
   for (j=1; j <= i² ; j++)
   sum[i] = sum[i] + j; // MyStatement
}
return true
```

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

$$= O\left(n^{\frac{3}{2}}\right) = O(n\sqrt{n}).$$

(b) (6 points) Express the time complexity of the following functions using big O() notation.

$f(n) = 9n^4 \log^3 n + \ 4n^5 + 4000n$

$$= O(n^5)$$

$g(n) = 1 / n$

$$= O(1)$$

$h(n) = 2^n \ + 500n^{200}$

$$= O(2^n)$$

(c) (4 points) Assume that we used a singly linked list to implement a stack, where the singly linked list has a reference to head only, i.e., it does not contain a reference to tail. What will the time complexity be of the **`push(el)`** and **`pop()`** operations, if applied on a singly linked list containing n elements? Justify your answer.

Both of them will cost O(1) since we will always insert (for push) and delete (for pop) from the head of the singly linked list, and both of these operations require constant time with the availability of the head reference.

Q.3: (20 points) Linked Lists:

(a) (8 points) Write assignment statement(s) required to transform the doubly linked list of Figure 1 to that of Figure 2. Note that the changes have been made **bold** for clarification.

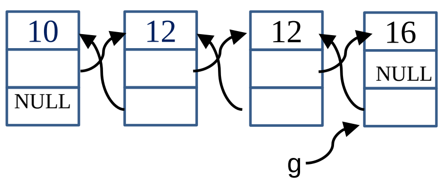


Figure 1

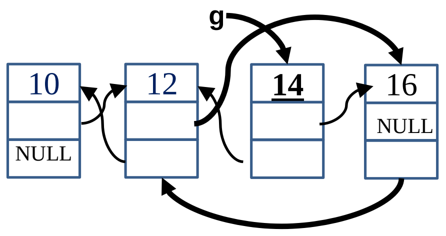
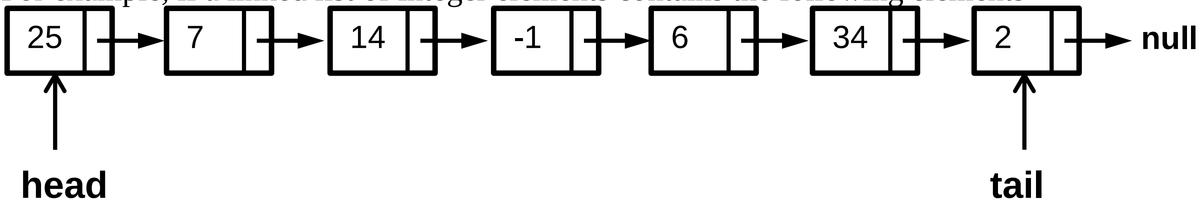


Figure 2

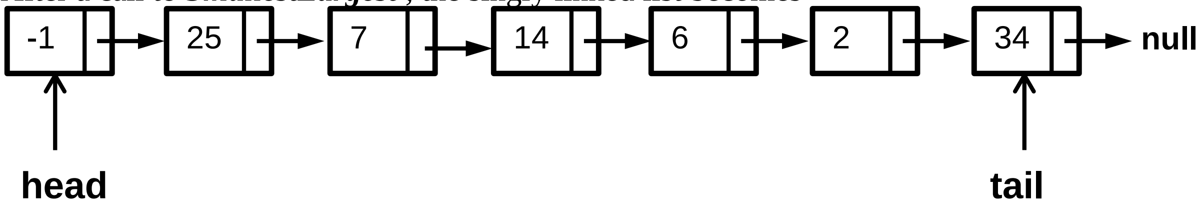
```
g.prev.info=14;
g.prev.prev.next=g;
g.prev=g.prev.prev;
g=g.prev;
```

(b) (12 points) Consider the SLL class definition shown in the reference sheet. Implement a method called ***SmallestLargest*** that places the minimum value in the list as the first element of the list and the largest value in the list as the last element in the list.

For example, if a linked list of Integer elements contains the following elements



After a call to ***SmallestLargest*** , the singly linked list becomes



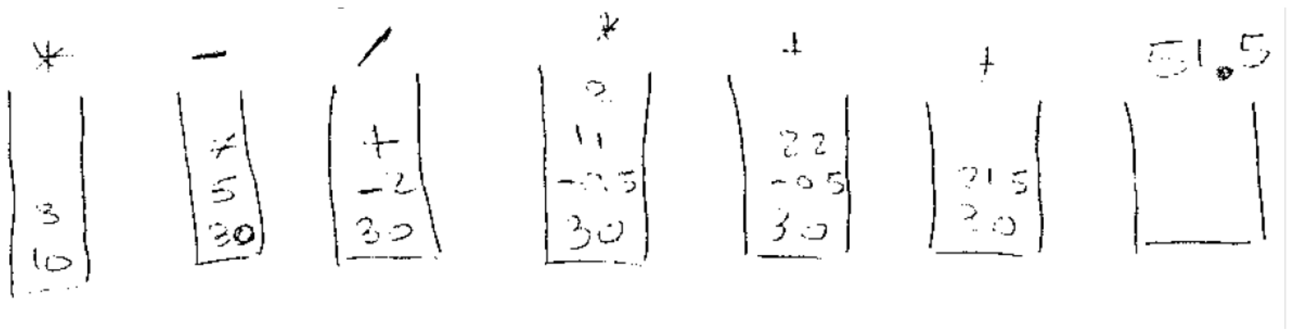
```
public void SmallestLargest(SLL<T> L) {
   // first, we check if the linked list is empty
  // or has one element
  if (L.isEmpty() || L.head==L.tail)
     return; // nothing needs to be done
  // now, we need to search for the smallest and largest
  SLLNode<T> r_temp; T smallest, largest;
  // we start by initializing smallest and largest to the
  // value in at the head of the linked list.
  r_temp = L.head; smallest = largest = r_temp.info;
  while (r_temp.next != NULL) {
     r_temp = r_temp.next;
     if (r_temp.info.compareTo(smallest)<0)
        smallest = r_temp.info;
     else if (r_temp.info.compareTo(largest)>0)
        largest = r_temp.info;
  }
  // Now, make sure that smallest and largest are different
  // If not, this means that all elements are equal and hence
  // nothing needs to be done.
  if (smallest != largest) {
    // delete smallest and insert it into the beginning of L
       L.delete(smallest);
       L.addToHead(smallest);

    // delete largest and insert it into the end of L
       L.delete(largest);
       L.addToTail(largest);
  }
}
```

Q.4: (20 points) Stacks and Queues:

(a) (8 points) Consider the following postfix expression:

10 3 * 5 7 – 4 / 11 2 * + +

(i) (6 points) Evaluate the postfix expression using a stack. Show the contents of the stack after each operation.



(ii) (2 points) Show the unambiguous infix equivalent of the above expression (you can use parentheses).

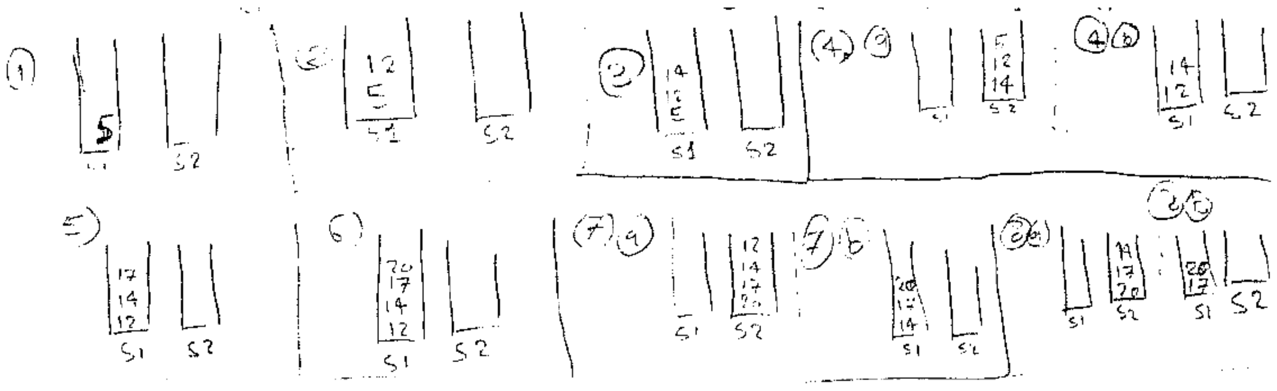
(10*3) + ((5-7)/4) + (11*2)

(b) (9 points) Implement the methods `enqueue(el)` and `dequeue()` for a queue, using two stacks, `S1` and `S2` with their standard operations (`push(el)`, `pop()`, `isEmpty()`).

```
void enqueue(T element) {
  s1.push(element);
}

T dequeue() {
  // transfer all elements of s1 into s2
  while (!s1.empty())
    s2.push(s1.pop());

  // pop and store the top element from s2
  T ret = s2.pop();

// transfer all elements of s2 back to s1
  while (!s2.empty())
    s1.push(s2.pop());
return ret;
}
```

(c) (3 points) Show the contents of the stacks `S1` and `S2` after carrying out the following operations using your implementation in (b), assuming that the "queue" was initially empty.

`Q.enqueue(5), Q.enqueue(12), Q.enqueue(14), Q.dequeue(),Q.enqueue(17), Q.enqueue(20), Q.dequeue(),Q.dequeue().`



Q.5 (20 points) Recursion:

(a) (10 points) Consider the following recursive function:

```
Public static int sumArray (int[ ] x, int index) {
    int sum = 0;
    if (index == x.length)return sum;
    else {
        sum += x[index];
        return sumArray(x,index + 1);
    }
}
```

(i) (2 points) What is the error in **`sumArray`** function?

You should not use local variable to store results.

(ii) (8 points) Fix the error in **`sumArray`** function?

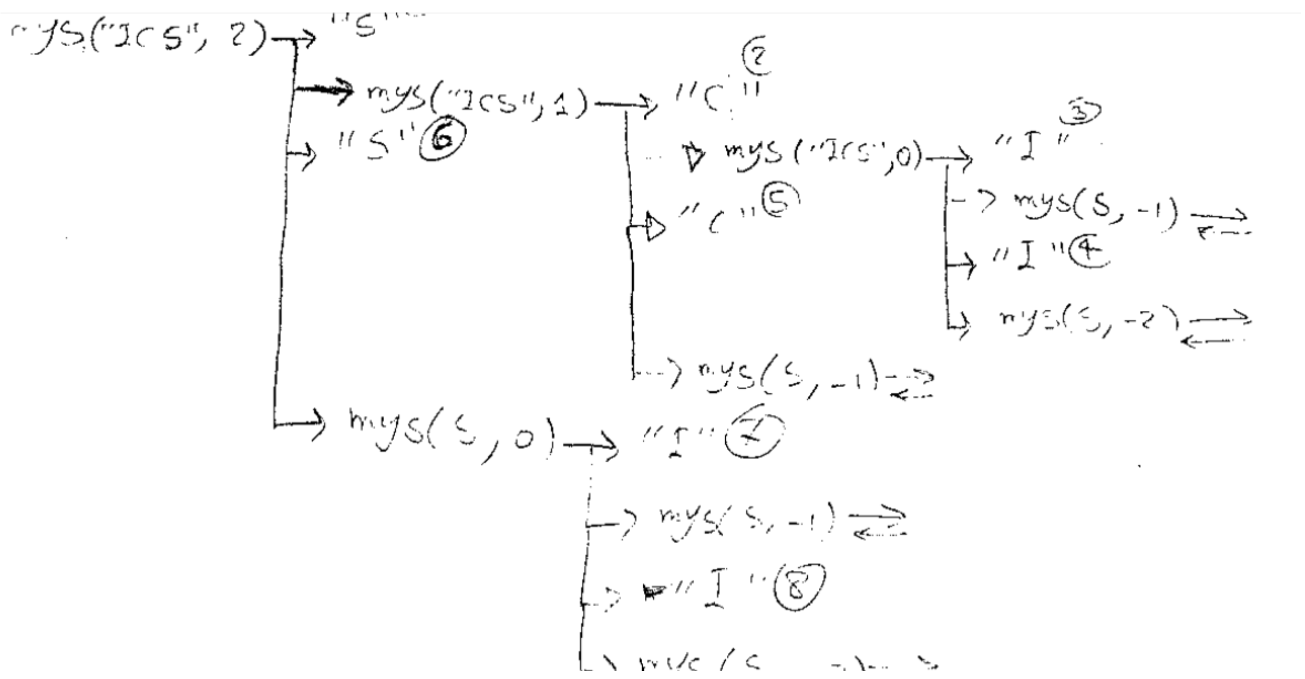
There are many solutions here to fix that.

(b) (10 points) Consider the following recursive method

```
public static void mysterious(String s, int index){
      if (index >= 0) {
         System.out.print(s.charAt(index));
         mysterious(s, index - 1);
         System.out.print(s.charAt(index));
         mysterious(s, index - 2);
    }
  }
```

Write down the recursion tree, representing the trace of the following method call, showing at the end the result returned by the call **`mysterious("ICS", 2).`**

S C I I S C I I



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();
}
```

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

$$\sum_{i=0}^{n} \frac{x^{n+1}-1}{x-1} \quad , \quad \log ab = \log a + \log b \quad , \quad \log_b a = \frac{\log_c a}{\log_c b}$$