Question 1 [64 marks] Each part is worth eight marks. Full marks for eight correct answers.

(i) Discuss briefly the following three concepts: ADT Stack, a Java interface for ADT Stack and a Java implementation for the ADT/interface, indicating clearly the relationships and distinctions among them. (8 marks)

- An ADT stack is a container capable of holding a number of objects, subject to a LIFO (last-in, first-out) discipline.
- A Java Interface is a class used for defining what elements (variables, methods) that must be implemented in any class that implements the interface.
- A Java Implementation for a java interface consists of implementing all methods and using all variables that have been previously defined in the java interface.

(ii) Give complete Java statements for each of the following. (a) Declare a variable named friends that is a list (ADT List) of Strings; (b) Create an empty list (using ArrayBasedList.java) and make friends refer to it; (c) add the three strings “tom”, “dick” and “harry” to the list (in the order shown); (d) remove the string “harry” and assign it to the String variable frnd. (8 marks)

```
List<String> friends = new ArrayBasedList<String>();
friends.add(“tom”);
friends.add(“dick”);
friends.add(“harry”);
String frnd = friends.remove(2);
```

(iii) Explain (using diagrams and/or prose as appropriate) the concept of a left-justified array as a basis for ADT implementations using ADT Map as an illustration. Sketch briefly (in words) how operation get might be implemented in terms of this representation. (8 marks)

- Answered in Summer 2009.

(iv) Complete the pseudocode for the array-based implementation for ADT Queue given below by providing code for placeholders alpha and beta. (8 marks)

```
Instance vars:
    Q[] array of length N
    f = 0, r = 0 ints

Algorithm isEmpty():
    return /* alpha */
```

```
Algorithm enqueue(o):
    if size () = N − 1 then
        copy queue contents into larger array
    Q[r] ← o
    r ← (r+1) mod N

Algorithm dequeue():
    if isEmpty() then
        issue error message
    /* beta */
```

alpha
return (f == r)

beta
temp = elements[f];
f = (f+1) % capacity;
return temp;

(v) The Java queue implementation given below includes a flawed version of the operation dequeue; the other methods are correct as shown. Identify as many defects as you can and suggest how they might be remedied. (8 marks)

```
public class LinkedQueue<EltType>
             implements Queue <EltType>
{  public LinkedQueue()
   {   head = null;
       tail  = null;
       size  = 0;
   }



   public void enqueue(EltType obj)
   {
       LLNode<EltType> node =
            new LLNode<EltType>();
       node.setElement(obj);
       node.setNext(null);
       if (size  == 0)
            head = node;
       else
            tail .setNext(node);
       tail  = node;
       size++;
   }
```

```
public EltType dequeue()
{
    EltType obj
    if (size  = 0)
       flagError(" illegal␣queue␣op");
    }
    obj = head.getElement();
    head = tail.getNext();
    size++;
    if (size  == 0)
       tail  = null;
    return obj;
}


 private LLNode<EltType> head;
 private LLNode<EltType> tail;
 private int size;
}
```

(vi) Give a complete Java fragment that for an array X of type String [] "expands" the array so at its conclusion X has twice the length it had originally but still contains all of the original elements. (8 marks)

```
String temp[] = (String[])(new Object[2*capacity]);
int indexIntoQ = f;
int indexIntoTemp = 0;

for (indexIntoQ = f; indexIntoQ != r;  indexIntoQ = (indexIntoQ +1) % capacity) {
            temp[indexIntoTemp] = elements[indexIntoQ];
            indexIntoTemp++;
}

f = 0;
r = indexIntoTemp;
elements = temp;
capacity = 2*capacity;
```

(vii) Illustrate the linear probing hashing technique by showing the table that results from the insertion of the following sequence of keys (in the order shown) into an initially empty hash table. Assume that the hash table has size 13 and that the hash function h(k) = k mod 13 is to be used. (8 marks)

1 5 3 7 14 27 40 53

(viii) Sketch the binary search tree that results from the insertion of the following sequence of keys into an initially empty tree.

4 6 2 1 7 5 3

Indicate the state of the tree after the deletion of key 4 and again after the reinsertion of key 4. (8 marks)

(ix) Give a complete pseudocode algorithm that for key k returns the node in the binary search tree bearing that key and returns the special value null if there is no such node. (8 marks)

(x) Discuss briefly how binary search might be exploited in the context of an array-based implementation of ADT Map and what benefit this might bring. Explain briefly any assumptions that you make. (8 marks)

- Using a map and an ADT Set may be used to exploit binary search
- Here, the map entry key could be the string 1-10, indicating that the map entry value was an ADT set consisting of the values 1-10
- This would allow for only one initial search e.g to find the number 17, you would get the value (ADT set) map entry with key 10-20.. followed by the normal binary search, but only within the specific set, greatly reducing the amount of search operations

(xi) A mode of a list of numbers is a value that appears with the greatest frequency. For example, the mode of 2, 1, 3, 2, 1, 2, 4, , 5 is the number 2. Outline an approach that computes a/the mode of a list efficiently. Your need not provide code or pseudocode, but both the feasibility and efficiency of your approach should be convincing. (8 marks)

- Create an empty map
- The map will be used to store the various numbers in the frequency as the key, and their frequencies as the value
- Iterate through the entire list, for each number
    - If it is present in the map (as a key) then get its value in the map and increment
    - Else, create a new map entry with key = to the number and value = 1
- Finally, iterate through the map to get the key with the greatest value

(xii) Describe an algorithm for the evaluation of simple arithmetic expression in postfix notation and illustrate the execution of the algorithm for the expression (8 marks)

1 2 + 3 * 4 −

- Use a scanner to get each separate element one by one e.g. elements 1, 2, +, 3 etc.
- Create a stack object, and variables called 'currentToken', 'firstOperand' & 'secondOperand'
- For each operand encountered while scanning, push it on the stack
- For each token encountered while scanning, pop last two operands off the stack
- Evaluate operands with token, and push result back onto stack
- Repeat until nothing left to scan

Question 2 [32 marks]

Give a complete Java implementation for ADT List. You must include a complete interface and a complete implementation, based either on a left-justified array or on a doubly linked list. For full marks your implementation must allow for elements of any type and must support all of the operations listed in the ADT summary appended at the end of this paper with the exception of operations iterator and listIterator, which you may ignore completely. Indicate clearly any assumptions you make and sketch briefly the behaviour of any ancillary classes that your implementation relies upon.

- Answered in Autumn 2010.

Question 3 [32 marks]

Discuss how each of the following three approaches may be used as a basis for the implementation of ADT Map: (i) a doubly-linked list, (ii) hashing with chaining and (iii)

a tree-based map. In each case, sketch how the map might be represented and how operation get might be implemented. You do not need to provide code, but your description (using diagrams and prose as appropriate) must be sufficiently detailed to address all the significant issues involved in providing a Java implementation based on the approach. Comment on the efficiency of operation get for each of the three approaches and indicate any other advantages/disadvantages or limitations of each approach.

- Not needed (Don't think we covered hashing with chaining or trees)

Question 4 [32 marks]

(i) Give a pseudocode description of the postorder tree traversal algorithm. (8 marks)

(ii) Prove that postorder applied to the root of a tree (ADT Tree) visits each node of tree. (12 marks)

(iii) Analyze the worst-case running time of the algorithm, stating any assumptions that you make. For the purposes of the analysis you may assume that the "visit" action can be completed in O(1) time. (12 marks)

Question 5 [32 marks]

Suppose that we wish to analyze the date-of-birth information of a group of students in order to identify clusters of students that share a common birthday.

Assume that classes Student and Date have been already been implemented. Date objects encode the day, month and year components of a specific date, while Student objects encode the name (assumed to be unique) and date of birth (as a Date) of an individual student. In each case objects support getter and setters methods for their constituent members.

Give a design for a Java application that takes student data (in the form of List<Student>) and that identifies clusters of students with a common birthday. In other words, for each for each cluster of two or more students sharing a birthday, the application should print the names of all the students in that cluster. Note that students sharing a birthday may have different dates of birth (if they are born in different years). Give a succinct outline of the algorithms and data structures employed by your application. (8 marks)

Specify in as much detail as you can, how the application is to be implemented. Full marks will be awarded for a complete and correct Java implementation; partial marks will be awarded in proportion to the degree to which your answer, using Java, pseudocode or prose as appropriate, correctly addresses the technical algorithmic, representational and programming issues that would be embodied in such a Java implementation. (24 marks)

You may assume that Java interfaces and implementations are available for all the ADTs listed on the sheet appended to this paper. State clearly any other assumptions you make about any other classes you employ.