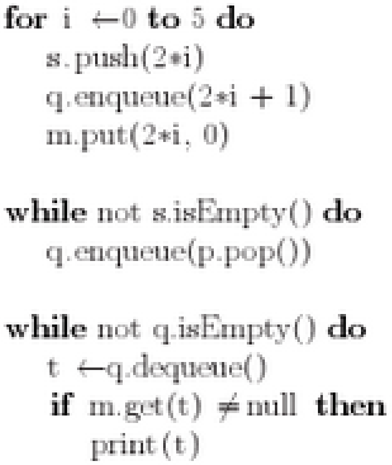
Question 1 [40 marks] Answer all eight parts.
(i) What output does the following code produce? Assume that s, q and m are instances of ADT Stack, Queue and Map, respectively and that all are initially empty. (5 marks)



```
for i ←0 to 5 do
    s.push(2*i)
    q.enqueue(2*i + 1)
    m.put(2*i, 0)

while not s.isEmpty() do
    q.enqueue(p.pop())

while not q.isEmpty() do
    t ←q.dequeue()
    if m.get(t) ≠ null then
        print(t)
```

- stack: 0, 2, 4, 6, 8, 10
- queue: 1, 3, 5, 7, 9, 11
- map: (10, 0)

- assume p.pop() is supposed to be s.pop()
- values 0, 2, 4, 6, 8, 10 are popped off stack and added to end queue to make (queue)1, 3, 5, 7, 9, 11, 0, 2, 4, 6, 8, 10

- for every element in the queue, if its in the map, print it
- Output: 10

(ii) Explain what is meant by the term left-justified array and describe how this concept may be used to represent ADT Stack. (5 marks)

- Answered in Summer 2012.

(iii) In terms of a left-justified representation of ADT Stack, give algorithms in pseudocode for operations push and pop. (5 marks)

- Answered in Summer 2012.

(iv) Describe, in words, diagrams or pseudo-code as appropriate, a suitable representation for ADT Map. (5 marks)

- Answered in Summer 2011 (double linked list representation).
- Answered in Summer 2009 (left-justified array).

(v) In terms of your chosen representation for ADT Map, give a complete pseudocode algorithm for operation remove. (5 marks)

- Answered in Summer 2009 (linked list).

(vi) Explain what a comparator is and why it is useful in the context of the Java implementation of ADT Map. (5 marks)

- Answered in Summer 2011.

(vii) Compare and contrast the linear search and binary search algorithms and comment on their respective worst-case running times. (5 marks)

- Linear search algorithm simply searches through an array of length n from 0 to arraySize - 1 for specified value x
    - Running time: 2n + 1
- Binary search algorithm searches for value x in array by using three specific variables low, mid, high, where mid = (low + high) / 2
    - Should the value being searched for be less than mid, high is set to mid - 1
    - Should the value being searched for be more than mid, low is set to mid + 1
    - This is done until mid == x (value being searched for)
    - Running time: 4+3 log2 n

(viii) Describe succinctly what a merging algorithm does when applied to two lists (ADT List). Give a complete pseudo-code implementation for the MergeSort sorting algorithm. (5 marks)

- Answered in Summer 2011.

Question 2 [20 marks]
ADT Deque is a queue-like ADT that supports insertions and deletions at both the front and the rear of the "queue". It supports the following main operations:

- **insertFirst(e): Insert a new element e at the beginning of the deque. Input: EltType; Output: None.**

- **insertLast(e): Insert a new element e at the end of the deque. Input: EltType; Output: None.**

- **removeFirst(): Remove and return the element at the beginning of the deque. Illegal if the deque is empty. Input: None; Output: EltType.**

- **removeLast(): Remove and return the element at the end of the deque. Illegal if the deque is empty. Input: None; Output: EltType.**

- **size(): Return the number of elements in the deque. Input: None; Output: int.**

- **isEmpty(): Return true if the dequeue is empty and false otherwise Input: None; Output: boolean.**

Give a suitable Java interface for ADT Deque. (4 marks)

Give suitable Java implementation of this ADT. For full marks your implementation must

- **be based on a linked-list representation representation of the ADT;**

- **be capable of handling any types of objects as elements;**

- **include suitable Java code for each of the following**

  - **instance variables (4 marks)**

  - **a constructor (4 marks)**

  - **implementations of operations insertFirst and insertLast (4 marks)**

  - **implementation of operations removeFirst and removeLast. (4 marks)**

Question 3 [20 marks]
Suppose we wish to write an application to analyze babies' names with a view to determining the most common choices. Each baby is represented by a Baby object that has the following members: firstName, lastName, pps Number and sex (all of type String), together with getters and setters for these. (A baby's sex is encoded as one of the strings "male" or "female").

(i) Give a Java fragment that takes an object babyList of type List<Baby> and that writes out the names (first name and last name) of each baby in the list. (4 marks)

```
List<Baby> babyList = new ArrayBasedList<Baby>();
Iterator<Baby> iterator = babyList.iterator();
while(iterator.hasNext()) {
        Baby b = iterator.next();
        System.out.println(b.firstName + " " + b.lastName);
}
```

(ii) Explain carefully how ADT Map might be used to record how many babies bear a particular name (first name). (4 marks)

- Create empty map
- Scan through list of babies using iterator
- For each baby
    - Get babies first name
    - Scan map for babies first name i.e. get(firstName)
        - If firstName is present in map, get value for key of the babies firstName
        - Increment its value by one
        - If firstName is not present, create new map entry where key is babies first name and value is 1

(iii) Write a Java algorithm that analyzes the contents of babyList and prints out whichever name (first name) is the most common (ignoring the possibility of ties). (8 marks)

```
// iterate babies
Map<String, Integer> map = new ArrayBasedMap<String, Integer>();
Iterator<Baby> iterator = babyList.iterator();
while(iterator.hasNext()) {
        Baby b = iterator.next();
        Integer nameCount = map.get(b.firstName);

        // baby name in list
        if(nameCount != null) {
                map.remove(b.firstName);
                map.put(b.firstName, (nameCount + 1));
        }
        // baby name not in list
        else {
                map.put(b.firstName, 1);
        }
}

// get most common baby name
Iterator<Entry<String, Integer>> mapIterator = map.iterator();
String name = "";
Integer nameCount = 0;

while(mapIterator.hasNext()) {
        Entry<String, Integer> entry = mapIterator.next();
        if(entry.getValue() > nameCount) {
                nameCount = entry.getValue();
                name = entry.getKey();
        }
}

System.out.println(name);
```

(iv) Describe briefly how to determine the ten most common boys' names. (4 marks)

- Use an iterator to iterate through each baby in the list of babies
- For each baby
    - If its male
        - Get its first and last name
        - Create a priority queue where its key will be the babies name and its value will be the amount of times this name occurs in the baby list
        - Check if this babies name is already in the priority queue
            - If so, get the value of the priority queue entry with the babies name
            - Increment its value by 1
            - Else, create a new priority queue entry with the key as the babies name and the value as 1
- Once finished iterating through entire baby list, simply create a loop that loops through the priority queue as long as there are more than 10 baby names, and uses the method removeMin() while doing so, which will give back the 10 most common boys names once the looping is finished.