Department of Computer Science and Engineering
York University, Winter 2009

CSE 2011: Assignment 1

Due Date - Monday, April 13, by Noon !

Question 1 **Algorithm Design** [10 points]

Assume an arbitrary set of n distinct numbers (**S**) distributed over an interval [0,N], n<<N. Devise an algorithm for outputting k smallest elements of **S** in order, where k<n. (Note: k is NOT a constant, but an easily computable function of n.) The number of 'compare' (comparison between two elements) operations in your algorithm should be O(n*log(k)) or better. The space requirements should not exceed O(n)[1], and only the use of linear data structures is allowed.

Give a pseudo-code description of your algorithm, and briefly justify its overall running time and the number of compare operations.

Question 2 **Big-O Analysis** [10 points]

(a) Express the running time of the following program segment using big-θ notation.

```
int k=1, sum=0;
for (int i=0; i<n; i++) {
   k=2*k; }
for (int j=k; j>1; j=j/3) {
   sum++; }
```

(b) Consider recursive method Enigma, as given below.

```
int Enigma(int num) {
    if (num > 4) { return 2 + Enigma(num - 5) + Enigma(num - 2);}
    return 1; }
```

What is the value returned from the call Enigma(5)?
What is the value returned from the call Enigma(15)?

Justify your answers, both in (a) and (b).

[1] This precludes the use of Radix/Bucket Sort algorithm. Hence, in the best case, it takes O(n·log(n)) to sort the given set.

Question 3 **ADTs** [10 points]

Suppose you are given a class **Queue** which implements the standard **enqueue**(E element) and **dequeue**() methods. The nature of the underlying data structure is not known.
Describe how you would use this class to implement **Stack** class. Specifically, explain how you would implement **Stack**'s **push**(E element) and **pop**() methods using **Queue**'s **enqueue**(E element) and **dequeue**().

Question 4 **Java Programming Project: On-line Message Assembler** [70 points]

When Bob wants to send Alice a message (M) on the Internet, he breaks M into n data packets, numbers the packets consecutively, and injects them into the network. When the packets arrive at Alice's computer, they are out of order, so Alice must sort the sequence of n packets in order before she is able to read the entire message.

You are asked to create a Java program, which would help Alice assemble Bob's messages. You can assume that for each message (M), the contents of n received packets, together with their respective sequence numbers, are temporarily stored in a file, as illustrated in Figure 1. An example of such a file, which can be used for test purposes, is available at:
http://www.cs.yorku.ca/course/2011/2011N/Mystery.txt

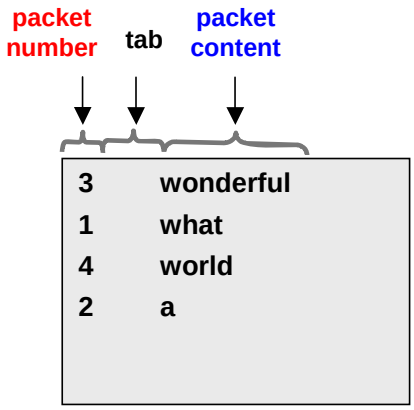

Figure 1 File content

Your program design should be based on the following guidelines:

(1) Create DLLNode[2] class, which will be used to store the content of each individual packet. The interface of this class is given below.

```
class DLLNode{
   DLLNode(int packetID, String packetContent, DLLNode prev, DLLNode next){}
   int getPacketID() { }
   String getPacketContent() { }
   DLLNode getNextNode() { }
```

[2] Here DLL stands for doubly linked list.

```
   DLLNode getPrevNode() { }
   void setNextNode(DLLNode n) { }
   void setPrevNode(DLLNode p) { }
}
```

(2) Create DLL class, which will be used to store the content of an entire file (i.e. message) – one line/packet per node. The outline of this class is given below. You are allowed to add new methods to this class, as needed.

```
class DLL {

   /* readAndAssemble performs the following operations:
      (1) Reads from file fileName (e.g. Mystery.txt) line by line
      (2) stores the content of each line in a DLLNode
      (3) places each DLLNode in DLL according to DLLNode's packetID number  */
   void readAndAssemble(String fileName) { }

   /* printContent traverses DLL and prints out the content of its nodes so as to
      recreate the original message */
   void printContent() { }

}
```

(3) Create MessageAssembler class, which will contain main() method and act as a tester class. The outline of this class is given below. You are allowed to add new methods to this class, as needed.

```
public class MessageAssembler {

   public static void main(String[] args) {

            DLL myDLL = new DLL();
            myDLL.readAndAssemble("Mystery.txt");
            myDLL.printContent();

      }
}
```