CpSc 374 Midterm Exam

Spring 2019

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. ( 30 pts) Determine the order of each of the six Queue operations/methods in terms of N (number of items in the Q). Include constant multipliers (e.g., 2N) in your answers. Use the blanks to provide your answers

**class Queue**

 {

 private **LinkList** myQ; //code provided below

 public **Queue**(int s) **// Queue() is O( )**

 {

 myQ = new LinkList;

 maxsize=s;

 }

 public void **insert**(long j) **// insert() is O( )**

 {

 int sz=myQ.size();

 long [] temp = new long(sz);

 int i = 0;

 while (!myQ.isEmpty()){

 temp[i]=myQ.deleteFirst();

 i++;

 }

 myQ.insertFirst(j);

 for (i=maxsize-1; i>=0; i--)

 myQ.insertFirst(temp[i]);

 **}**

 public long **remove**() **// remove() is O( )**

 {

 if (!myQ.isEmpty())

 return myQ.deleteFirst();

 }

 public long **peekFront**() **// peekFront() is O( )**

 {

 if (myQ.isEmpty())

 return -1;

 long retVal = myQ.deleteFirst();

 myQ.insertFirst(retVal));

 return retVal;

 }

 public boolean **isEmpty**() **// isEmpty() is O( )**

 {

 return (myQ.isEmpty());

 }

 public int **size**() **// size() is O O( )**

 {

 Node temp;

 int count = 0;

 temp=myQ.first;

 while (temp.next != null){

 temp=temp.next;

 count++;

 }

 return count;

 }

 }

1. ( 5 pts) What is the space complexity of the **insert**(long j) method. Include constant multipliers in your answer. EXPLAIN your answer!

**Thus, insert() Space Complexity is O( )**

1. (15 pts) Write an algorithm for an improved version of the **insert**(long j) method. Analyze your improved method, provide its O() running time and explain how you arrived at that answer.

//---------------------- this part is for reference only -----------

class Node

 {

 public long dData;

 public Node next;

 public Node(long d)

 { dData = d; }

 public void displayNode()

 { System.out.print(dData + " "); }

 } // end class Node

class LinkList

{

 public Node first;

 public LinkList()

 { first = null; }

 public boolean isEmpty()

 { return (first==null); }

 public void insertFirst(long dd)

 {

 Link newLink = new Link(dd);

 newLink.next = first;

 first = newLink;

 }

 public long deleteFirst()

 {

 Link temp = first;

 first = first.next;

 return temp.dData;

 }

 public void displayList()

 {

 Link current = first;

 while(current != null)

 {

 current.displayLink();

 current = current.next;

 }

 System.out.println("");

 }

} // end class LinkList