作业1

1. 答：

public class AssignmentOfTheory {  
 public static void main(String[] args) {  
 for (int i = 1; i < 100; i+=4) {  
 System.*out*.println(*func1*(i)>*func2*(i));  
 }  
 }  
 private static int func1(int n){  
 return 100\*n\*n;  
 }  
 private static int func2(int n){  
 return (int) Math.*pow*(2,n);  
 }  
}

前四个是true，即1，5，9，13是true，改变程序为：

public class AssignmentOfTheory {  
 public static void main(String[] args) {  
 for (int i = 1; i <= 15; i++) {  
 System.*out*.println(*func1*(i)>*func2*(i));  
 }  
 }  
 private static int func1(int n){  
 return 100\*n\*n;  
 }  
 private static int func2(int n){  
 return (int) Math.*pow*(2,n);  
 }  
}

发现i=14是true，15是false，因此是n<=14时后一个比前一个短。

作业2

1. 答：

Z=x+y是7500次

Z=x-y是2500/4=625

2. 答：

(1)n次

(2)n(n+1)(n+2)/6 次

作业3

1. 答：.A最小，因为logn<n<n^i(i>0)

2. 答：使用类似于二分查找的方法即可

public int eleSearch(int[] arr){  
 int low = 0;  
 int high = arr.length-1;  
 int center ;  
 while (low <= high) {  
 center = low+((high-low)>>>1);  
 if(arr[center] == center){  
 return arr[center];  
 }else if(arr[center]<center){  
 low = center+1;  
 }else if(arr[center]>center){  
 high = center-1;  
 }  
 }  
 return -1;  
}

作业4

1. 答：使用快速幂算法计算实数的整数幂即可

用自然语言解释就是寻找离幂次数最近的2的整数倍，然后2的整数倍这一部分可以二分，

其他的部分乘上去就行。需要注意的是二分并不一定对所有情况都是最佳的，但确实是一个最通用且较为快速的办法。

程序代码如下：

public class FastPow {  
 public static double fastPow(double x,int n){  
 double ans = 1;  
 long y = n;  
 if(n<0){  
 y=-y;  
 x=1/x;  
 }  
 while(y>0){  
 if(y%2==1){  
 ans\*=x;  
 }  
 x=x\*x;  
 y/=2;  
 }  
 return ans;  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*fastPow*(7,65));  
 }  
}

2. 答：(1)(4)(6)可以抽象为线性表

(2)是树形结构，(3)是哈希表或者称为集合，(5)是图

作业5

1. 答：我认为尾插最好应该提供，头插可以不提供。对于删除方法也是相同答案。

这主要原因是头插可以通过insert(0)来轻松实现，而尾插需要使用到线性表的长度，如果让用户额外调用一次求长度会显得十分繁琐。因此应该把尾插封装成一个方法，在指针遍历到最后一个结点时直接进行插入。对于删除方面，理由与以上说法相似。

2. 答：

loop（遍历）,contains（是否包含某元素(返回布尔值)）,remove,insert

作业6

1. 答：

先描述清楚节点类：

static class Node{  
 public Node(int value, Node next) {  
 this.value = value;  
 this.next = next;  
 }  
  
 int value;  
 Node next;   
}

一般的算法：

public boolean notRepeated(Node head){  
 Node p = head;  
 Node q = head.next;  
 while(p.next!=null){  
 p = p.next;  
 while(q.next!=null){  
 q = q.next;  
 if(p.value == q.value){  
 return false;  
 }  
 }  
 }  
 return true;  
}

使用哈希表的算法：

public boolean notRepeatedWithHash(Node head){  
 HashSet<Integer> hash = new HashSet<>();  
 Node p = head;  
 while(p!=null){  
 if(hash.contains(p.value)){  
 return false;  
 }  
 hash.add(p.value);  
 p = p.next;  
 }  
 return true;  
}

以上方法当不重复时返回true，有重复时返回false。

\*另外，使用哈希表的方法没有手写数组模拟哈希表的原因是我们不清楚链表的数据范围。

2. 答：

//取中间位序元素(快慢指针法)  
public int[] getMidValue(Node head){  
 Node fast = head;  
 Node slow = head;  
 int step = 0;  
 while(fast.next!=null){  
 fast = fast.next;  
 step++;  
 if(step%2==0){  
 slow = slow.next;  
 }  
 }  
 if(step%2==0){  
 return new int[]{slow.value,slow.next.value};  
 }else{  
 return new int[]{slow.value};  
 }  
}

返回值是一个数组，如果原链表的长度为偶数，则会返回中间两个数的值，如果是奇数返回中间一个数的值。

作业7

1. 答：

(1) ×

(2) ×（顺序表取第i个元素永远是O(1),链表查询效率才与i有关）

(3)√

(4)√

(5)√

2. 答：使用双向链表即可，带头哨兵和尾哨兵。这样头插尾插、头删尾删的效率最高。

这也是JDK自带LinkedList默认的实现方式